# Continuous Latent Variables

*Prof. Nicholas Zabaras*
*Center for Informatics and Computational Science*
*https://cics.nd.edu/*
*University of Notre Dame*
*Notre Dame, Indiana, USA*

*Email: nzabaras@gmail.com*
*URL: https://www.zabaras.com/*

*March 21, 2019*

# Contents - PCA

- Continuous Latent Variable Model, Generative Point of View, Factor Analysis, Unidentifiability, Mixture of Factor Analysers
- Principal Component Analysis, Maximum variance formulation, Minimum-error formulation, PCA Reconstruction , PCA and SVD
- Applications of PCA, Off-line Digit Images, Whitening of the data with PCA , Old Faithful Data Set, Oil Flow Data Set, PCA for Visualization, PCA Vs. Fisher Discriminant
- PCA for High-Dimensional Data

Following closely Chris Bishops' PRML book (Chapter 12)

# Contents – Probabilistic PCA

- Probabilistic PCA, Generative View Point, Predictive Distribution, Posterior Distribution, Maximum likelihood PCA, Efficient Evaluation of PPCA, Maximum Likelihood PCA Vs. Standard PCA, Probabilistic PCA – Degrees of Freedom

- EM algorithm for PCA, EM Cost, EM Missing Values, EM for $\sigma^2 \rightarrow 0$, EM Examples, EM Algorithm and Standard PCA Revisited

Following closely Chris Bishops' PRML book (Chapter 12)

# Contents-Nonlinear Latent Variable Models

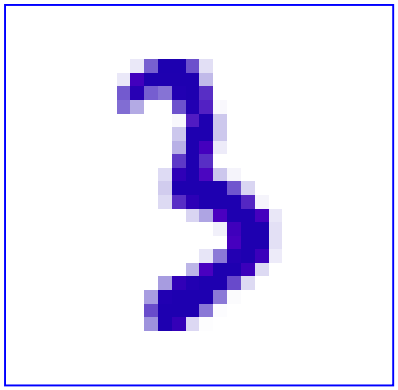
- Independent component analysis, Fast ICA Algorithm, Modeling the Source Densities, EM Implementation, Other Estimation Techniques, Maximizing non-Gaussianality, Maximizing Mutual Information
- Autoassociative neural networks
- Modelling nonlinear manifolds, Mixture of PCA, Principal Curces, Multidimensional Scaling, Locally Linear Embedding, ISOMAP, Latent Trait Models, Density Network
- Generative Topographic Mapping

Following closely Chris Bishops' PRML book (Chapter 12)

# Continuous Latent Variables - Introduction

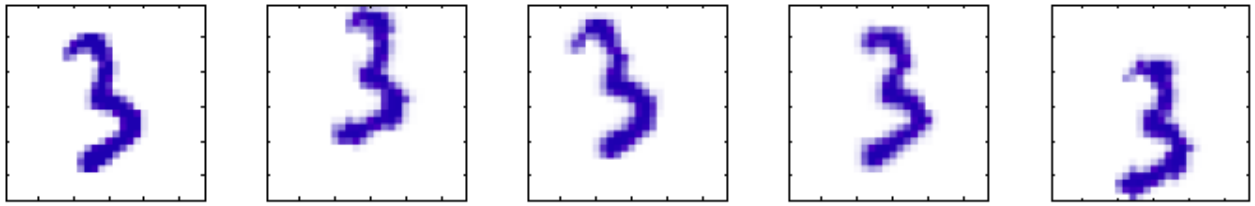
# Continuous Latent Variables

- In many data sets, the data points lie close to a manifold of much lower dimensionality than that of the original data space.
- Consider a data set constructed by taking one of the off-line digits, represented by a $64 \times 64$ pixel grey-level image.
- Embed this in a larger image of size $100 \times 100$ by padding with pixels having the value zero (white pixels).
- Each image is represented by a point in $10{,}000$ dimensional space.



# Continuous Latent Variables

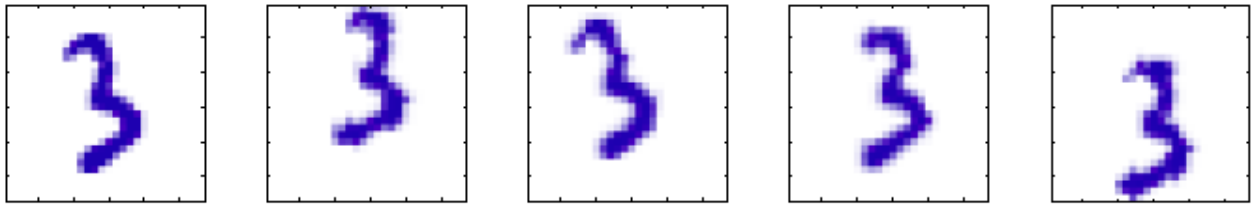
- We create multiple copies in which the location and orientation of the digit is varied at random.



- In this data set, there are 3 DOF of variability:
  - Vertical displacement
  - Horizontal displacement and
  - Rotation

# Intrinsic Dimensionality of a Data Set

- The intrinsic dimensionality of the data set is three (vertical and horizontal displacement and rotation)



- The manifold is non-linear: when translating the digit past a particular pixel, that pixel value goes from 0 (white) to 1 (black) and back to zero again. This is a nonlinear function of the digit position.

- The translation and rotation parameters are latent variables: we observe only the image vectors without knowing the translation or rotation variables used to create them.

# Other Latent Variables

- For real digit image data, additional DOF arise from
  - Scaling
  - Complex deformations due to the variability of an individual's writing style,
  - Etc.
- The number of such degrees of freedom will still be small compared to the dimensionality of the data set.
- For data compression, we are interested to explore the manifold structure.

# Generative Point of View

# Generative Point of View

- The data points are often not confined precisely to a smooth low-dimensional manifold.
- The departure of data points from the manifold is interpreted as 'noise'.
- **Generative view:**
  - (a) Select a point within the manifold according to some latent variable distribution $p(\boldsymbol{z})$ and
  - (b) Generate an observed data point by adding noise, drawn from some conditional distribution of the data variables given the latent variables $p(\boldsymbol{x}|\boldsymbol{z})$.

# Generative Point of View

- **Example – Linear Gaussian latent variable model:**
  - (a) assume Gaussian distributions for the latent variables $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$ and the observed variables $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$
  - (b) make use of a linear-Gaussian dependence of the observed variables on the state of the latent variables, $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$
- This leads to a probabilistic formulation of principal component analysis (PCA).

# Generative Point of View

## Example – Factor Analysis:

- (a) assume Gaussian distributions for the latent variables

$$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_0, \boldsymbol{\Sigma_0})$$

and the observed variables

$$p(\boldsymbol{x}_i|\boldsymbol{z}_i, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi}),$$
$$\boldsymbol{x}_i \in \mathbb{R}^D, \boldsymbol{z}_i \in \mathbb{R}^M, \boldsymbol{W} \in \mathbb{R}^{D\times M}, \boldsymbol{\Psi} \in \mathbb{R}^{D\times D}$$

- (b) We take $\boldsymbol{\Psi}$ to be diagonal. This overall model is called **factor analysis** or **FA**.

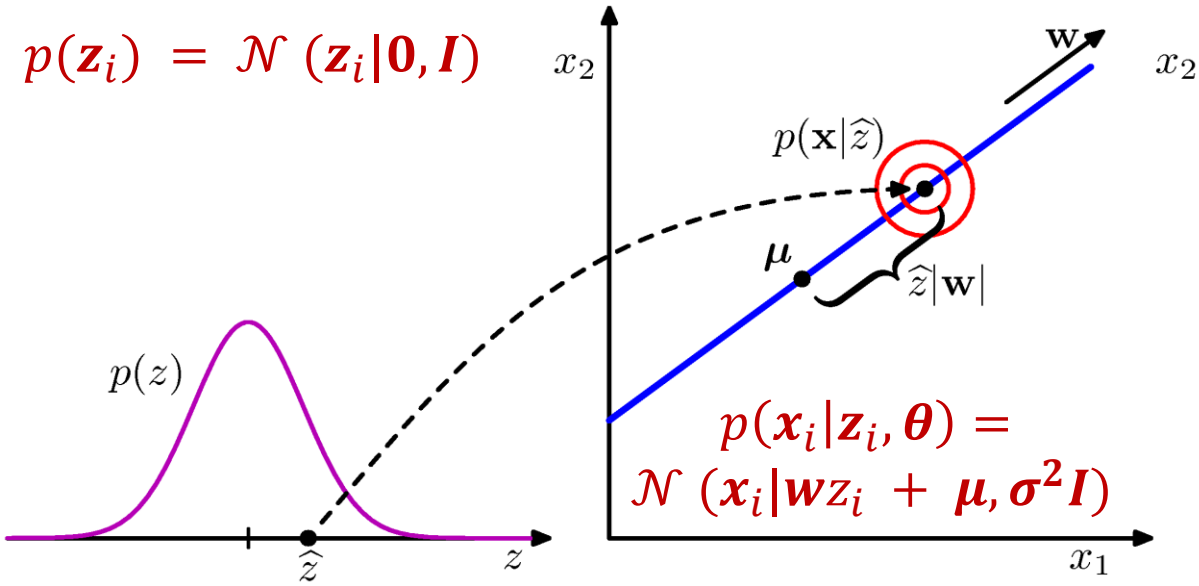
- The special case in which $\boldsymbol{\Psi} = \sigma^2\boldsymbol{I}$ is called **probabilistic principal components analysis** or **PPCA**.

# Generative Point of View

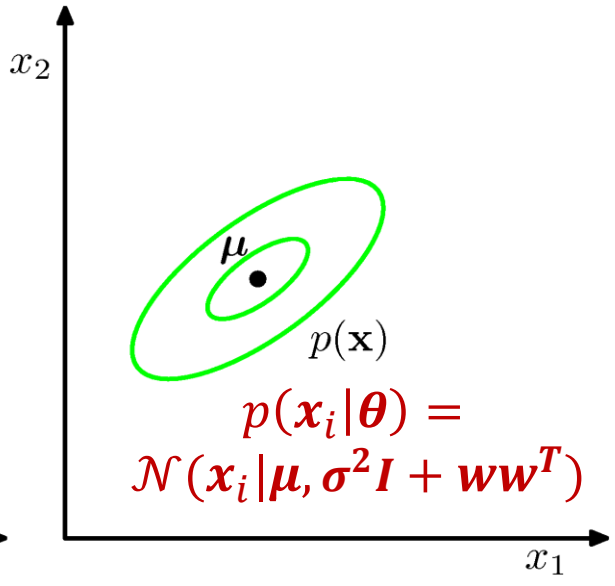
- PPCA is illustrated for $D = 2, M = 1$ (latent dimension).

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi})\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)d\boldsymbol{z}_i = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{W}\boldsymbol{\mu}_0 + \boldsymbol{\mu}, \boldsymbol{\Psi} + \boldsymbol{W}\boldsymbol{\Sigma}_0\boldsymbol{W}^T)$$

$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{0}, \boldsymbol{I})$




$p(\boldsymbol{x}_i|\boldsymbol{z}_i, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$




$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}, \sigma^2\boldsymbol{I} + \boldsymbol{w}\boldsymbol{w}^T)$

$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ $p(\boldsymbol{x}_i|\boldsymbol{z}_i, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi})$ $p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{W}\boldsymbol{\mu}_0 + \boldsymbol{\mu}, \boldsymbol{\Psi} + \boldsymbol{W}\boldsymbol{\Sigma}_0\boldsymbol{W}^T)$

- Based on the obtained marginal, we see that without sacrificing generality, we can simplify as:

$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{0}, \boldsymbol{I})$

$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Psi} + \boldsymbol{W}\boldsymbol{W}^T)$

# Factor Analysis

# Factor Analysis

- FA is a low rank parameterization of a MVN.
- FA can be thought of as a way of specifying a joint density model on $\boldsymbol{x}$ using a small number of parameters

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\boldsymbol{x}_i|\mathbf{W}\boldsymbol{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi})\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_\mathbf{0})d\boldsymbol{z}_i$$
$$= \mathcal{N}(\boldsymbol{x}_i|\mathbf{W}\boldsymbol{\mu}_0 + \boldsymbol{\mu}, \boldsymbol{\Psi} + \mathbf{W}\boldsymbol{\Sigma}_\mathbf{0}\mathbf{W}^T)$$

- From this, we see that we can set $\boldsymbol{\mu}_0 = 0$ without a loss of generality.
  - we can always absorb $\mathbf{W}\boldsymbol{\mu}_0$ into $\boldsymbol{\mu}$.
- Similarly, we can set $\boldsymbol{\Sigma}_\mathbf{0} = \mathbf{I}$.
  - we can always emulate a correlated prior by defining a new weight matrix $\widetilde{\mathbf{W}} = \mathbf{W}\boldsymbol{\Sigma}_0^{\frac{1}{2}}$.
- Therefore,

$$\text{cov}[\boldsymbol{x}|\boldsymbol{\theta}] = \mathbf{W}\boldsymbol{\Sigma}_\mathbf{0}\mathbf{W}^T + \boldsymbol{\Psi} = \left(\widetilde{\mathbf{W}}\boldsymbol{\Sigma}_0^{-\frac{1}{2}}\right)\boldsymbol{\Sigma}_\mathbf{0}\left(\widetilde{\mathbf{W}}\boldsymbol{\Sigma}_0^{-\frac{1}{2}}\right)^T + \boldsymbol{\Psi} = \widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^T + \boldsymbol{\Psi}$$

# Factor Analysis

- We thus see that FA approximates the covariance matrix of the visible vector using a low-rank decomposition:

$$\mathbf{C} \triangleq \text{cov}[\boldsymbol{x}] = \mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi}$$

- $\mathbf{W}$ only uses $\mathcal{O}(MD)$ parameters, which allows a flexible compromise between a full covariance Gaussian with $\mathcal{O}(D^2)$ and a diagonal covariance with $\mathcal{O}(D)$ parameters.

- Note that if we did not restrict $\boldsymbol{\Psi}$ to be diagonal, we could trivially set $\boldsymbol{\Psi}$ to a full covariance matrix.

  Then we can set $\mathbf{W} = \mathbf{0}$. In this case, latent factors will not be required.

# Inference of Latent Factors

- Although FA can be thought of as just a way to define a density on $\boldsymbol{x}$, it is often used because we hope that the latent factors $\boldsymbol{z}$ will reveal something interesting about the data.

- To do this, we need to compute the posterior over the latent factors.

- We can use Bayes' rule for Gaussians to give

$$p(\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{m}_i, \boldsymbol{\Sigma}_i)$$
$$\boldsymbol{\Sigma}_i \triangleq (\boldsymbol{\Sigma}_0^{-1} + \mathbf{W}\boldsymbol{\Psi}^{-1}\mathbf{W})^{-1}$$
$$\boldsymbol{m}_i \triangleq \boldsymbol{\Sigma}_i(\mathbf{W}^T\boldsymbol{\Psi}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}) + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0)$$

- Note that in the FA model, $\boldsymbol{\Sigma}_i$ is actually independent of $i$. Computing this matrix takes $\mathcal{O}(M^3 + M^2D)$ time.

- Computing each $\mathbf{m}_i = \mathbb{E}[\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{\theta}]$ takes $\mathcal{O}(M^2 + MD)$ time.

- The $\mathbf{m}_i$ are called latent score or latent factors.

# Inference of Latent Factors

- Use

$$p(\boldsymbol{x}_i|\boldsymbol{z}_i,\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\mu},\boldsymbol{\Psi}),$$

$$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_0,\boldsymbol{\Sigma_0}),$$

Bayes' rule and standard eqs. for linear Gaussian systems to derive the posterior of the latent variables:

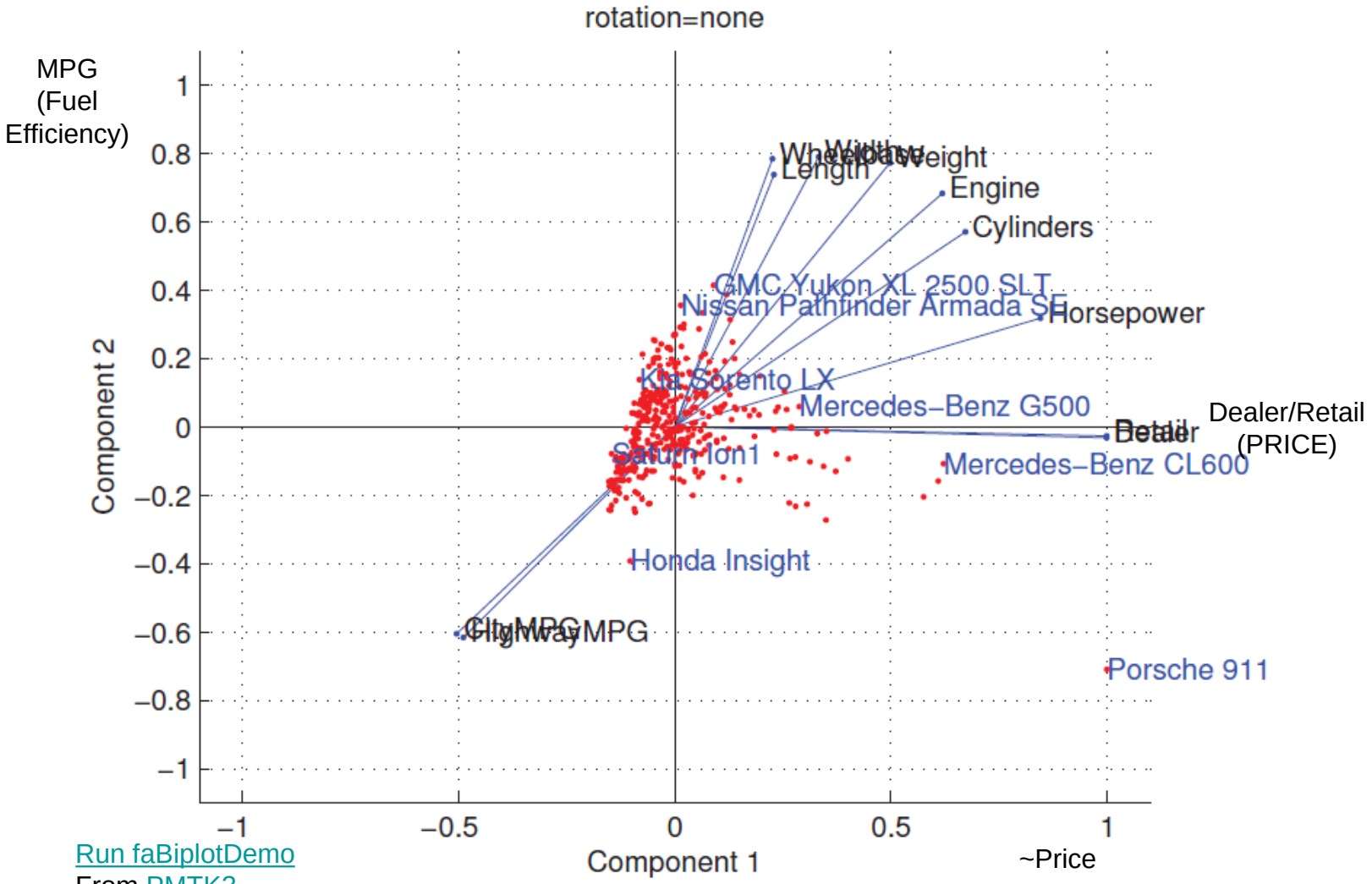
$$p(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{m}_i,\boldsymbol{\Sigma}_i),$$
$$\boldsymbol{m}_i = \boldsymbol{\Sigma}\left(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}(\boldsymbol{x}_i - \boldsymbol{\mu}) + \boldsymbol{\Sigma}_{\mathbf{0}}^{-\mathbf{1}}\boldsymbol{\mu}_0\right)$$
$$\boldsymbol{\Sigma} = \left(\boldsymbol{\Sigma}_{\mathbf{0}}^{-\mathbf{1}} + \boldsymbol{W}^{-\mathbf{1}}\boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{W}\right)^{-\mathbf{1}}$$

# Example Based on Factor Analysis

- Consider a dataset of $D = 11$ variables and $N = 387$ cases describing aspects of cars (engine size, # of cylinders, MPG, etc. ).
- We fit a $M = 2$ dimensional model. We plot the $\boldsymbol{m}_i = \mathbb{E}[\boldsymbol{z}_i | \boldsymbol{x}_i, \boldsymbol{\theta}]$ scores as points in $\mathbb{R}^2$, to visualize the data.
- We also project unit vectors corresponding to each of the feature dimensions, $\boldsymbol{e}_1 = (1, 0, \ldots, 0), \boldsymbol{e}_2 = (0, 1, 0, \ldots, 0)$, etc. into the low-dimensional space (blue lines).
- We see that the horizontal axis represents price corresponding to the features labeled “dealer” and “retail” (expensive cars on the right).
- The vertical axis represents fuel efficiency (in terms of MPG) versus size: heavy vehicles are less efficient and are higher up, whereas light vehicles are lower down.
- Unfortunately, interpreting latent variable models is fraught with difficulties.

# 2D Projection of Car Data

- Shalizi, C. (2009). Cs 36-350 lecture 10: Principal components: mathematics, example, interpretation.




Run faBiplotDemo
From PMTK3

# Unidentifiability

# Unidentifiability

- Consider an arbitrary rotation $\boldsymbol{R}$ such that: $\widetilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{R}$.
- We can immediately see that the likelihood function $p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{x}_i \middle| \boldsymbol{\mu}, \boldsymbol{\Psi} + \boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}}\right)$ remains the same:

$$\widetilde{\boldsymbol{W}}\,\widetilde{\boldsymbol{W}}^{T} = \boldsymbol{W}\boldsymbol{R}\boldsymbol{R}^{\boldsymbol{T}}\boldsymbol{W}^{\boldsymbol{T}} = \boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}}$$

- We need to remove $M(M-1)/2$ DOF since that is the number of orthonormal matrices of size $M \times M$.
- Then the FA model has $D + MD - M(M-1)/2$ free parameters (excluding the mean), where the first term arises from $\boldsymbol{\Psi}$. This needs to be $\leq D(D+1)/2$, i.e. the number of parameters in an unconstrained symmetric covariance matrix.
- This gives us an upper bound on $M$: $M_{max} = \lfloor D + 0.5\left(1 -$

# Unidentifiability

To address the unidentifiability the following options are available:

- Making $\boldsymbol{W}$ orthonormal: This is the case with PCA
  - columns arranged in order of decreasing variance
- Making $\boldsymbol{W}$ lower triangular: The first feature is then generated only by the 1st latent variable, the second by the first two, etc.
  - The number of parameters is equal to $M_{max}$.
  - One needs to properly select the first $M$ visible variables as they affect the latent factors.


- Lopes, H. and M. West (2004). Bayesian model assessment in factor analysis. *Statisica Sinica 14*, 41– 67.

# Unidentifiability

- Sparsity promoting priors on the weights: Use sparse factor analysis that forces some entries in $\boldsymbol{W}$ to zero. They may include $\ell_1$regularization, automatic relevance determination or spike and slab priors. But note that this does not ensure a unique MAP estimate.

- Choosing an informative rotation matrix $\boldsymbol{R}$: Choosing appropriate $\boldsymbol{R}$ to increase interpretable sparse $\boldsymbol{W}$.

- Using non-Gaussian priors for the latent factors in $p(\boldsymbol{z}_i)$: This leads to Independent Component Analysis (ICA).


- Zou, H. (2006). The adaptive Lasso and its oracle properties. *J. of the Am. Stat. Assoc.*, 1418–1429.
- Bishop, C. (1999). Bayesian PCA. In *NIPS*.
- Archambeau, C. and F. Bach (2008). Sparse probabilistic projections. In *NIPS*.
- Kaiser, H. (1958). The varimax criterion for analytic rotation in factor analysis, *Psychometrika 23*(3).

# Mixtures of Factor Analysers

# Mixtures of Factor Analysers

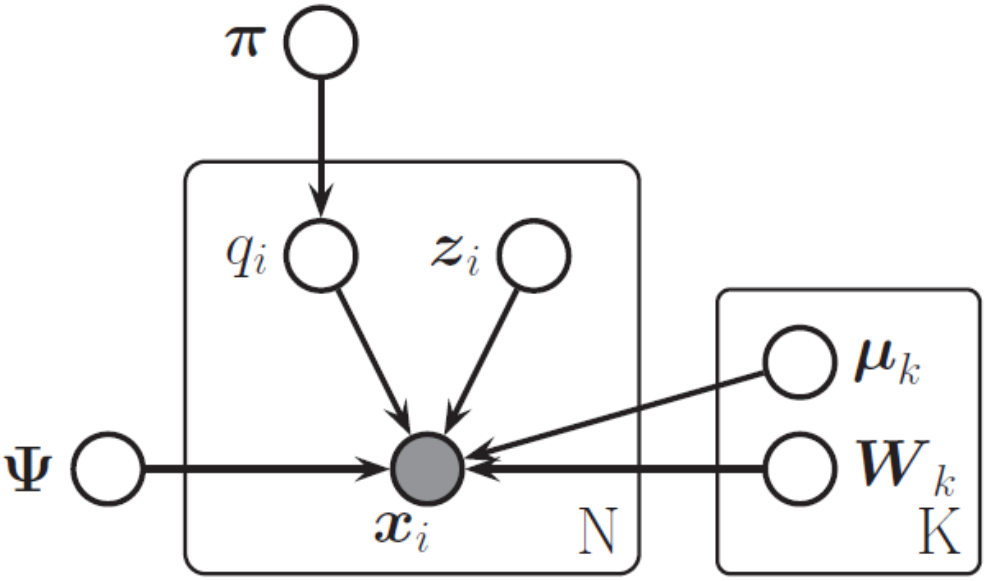
- The FA model assumes that the data lives on a low-dimensional linear manifold.
- In reality, most data is better modeled by some form of low-dimensional curved manifold.
- We can approximate a curved manifold by a piecewise linear manifold
  - This suggests the following model. Let the $k$-th linear subspace of dimensionality $\mathrm{M}_k$ be represented by $\mathbf{W}_k$, for $k = 1, \ldots, K$.
  - Suppose, we have a latent indicator $q_i \in \{1, \ldots, K\}$ specifying which subspace we should use to generate the data.
  - We then sample $z_i$ from a Gaussian prior and pass it through the $\mathbf{W}_k$ matrix and add noise:

$$p(\boldsymbol{x}_i|\boldsymbol{z}_i, q_i = k, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k + \mathbf{W}_k\boldsymbol{z}_i, \boldsymbol{\Psi})$$
$$p(\boldsymbol{z}_i|\boldsymbol{\theta}) = \mathcal{N}(z_i|\mathbf{0}, \mathbf{I})$$
$$p(q_i|\boldsymbol{\theta}) = \mathcal{C}at(q_i|\boldsymbol{\pi})$$

- This is called mixture of factor analysers.

# Mixtures of Factor Analysers

- A probabilistic graphical model of the mixture of factor analysers is given below.

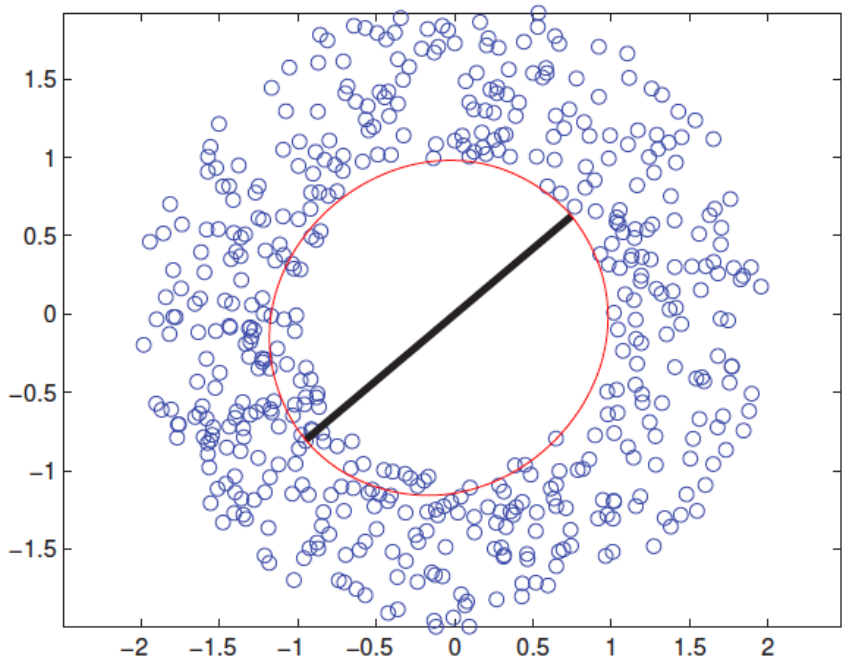


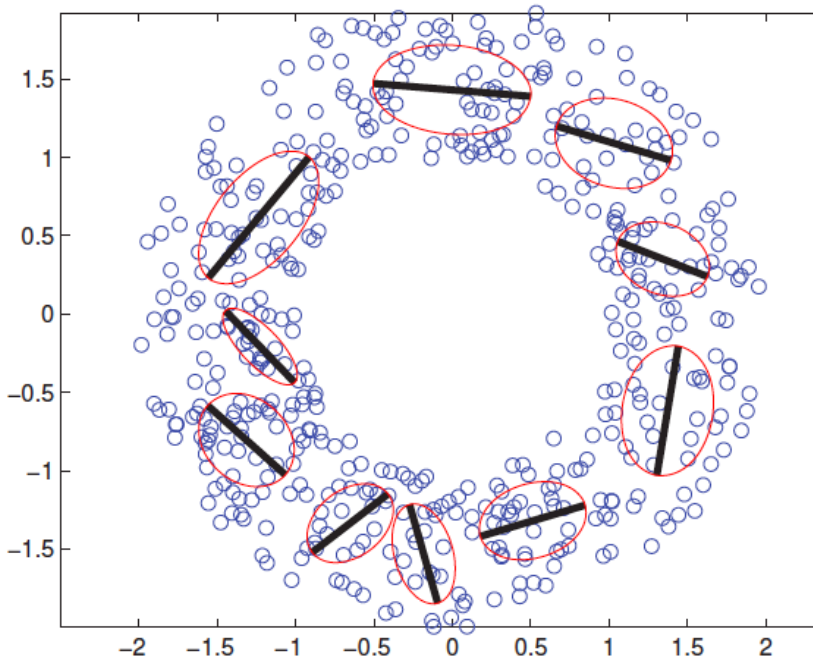

$$p(\boldsymbol{x}_i|\boldsymbol{z}_i, q_i = k, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k + \mathbf{W}_k\boldsymbol{z}_i, \boldsymbol{\Psi})$$
$$p(\boldsymbol{z}_i|\boldsymbol{\theta}) = \mathcal{N}(z_i|\mathbf{0}, \mathbf{I})$$
$$p(q_i|\boldsymbol{\theta}) = \mathcal{C}at(q_i|\boldsymbol{\pi})$$

# Mixtures of Factor Analysers

- Mixture of 1d PPCAs fit to a dataset, for $K = 1, 10$.

mixPpcaDemoNetlab
from PMTK3




- Think of this is as a low-rank version of a mixture of Gaussians
- The model needs $\mathcal{O}(KMD)$ parameters instead of the $\mathcal{O}(KD^2)$ parameters needed for a mixture of full-covariance Gaussians.
- This can reduce overfitting and thus MFA is a good generic density model for high-dimensional real-valued data.

# Latent Models

- PCA arises as the Maximum Likelihood solution to a linear Gaussian Latent Variable Model.
- Probabilistic formulation allows:
  - Use of EM for parameter estimation
  - Using mixtures of PCA models
  - A Bayesian formalism in addition allows computing the number of principal components directly from the data.
- Non Gaussian latent variable models lead to Independent Component Analysis.
- Models can also be considered with a non-linear relation between latent and observed variables.

# Principal Component Analysis

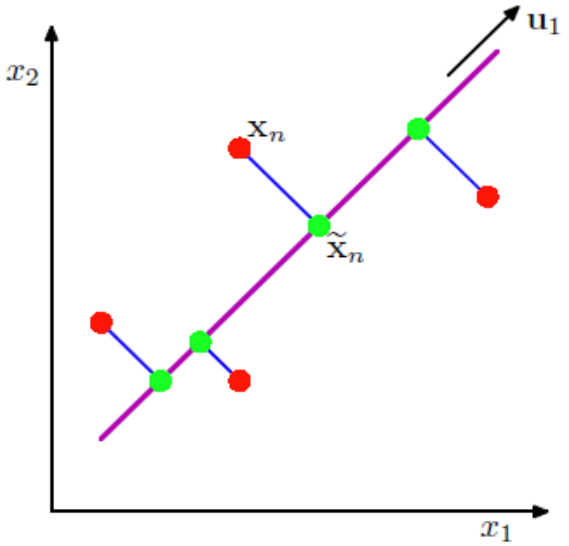
# Principal Component Analysis

- Principal component analysis (PCA) – also known as Karhunen-Loeve expansion - is used widely for (Jolliffe, 2002)
  - Dimensionality reduction
  - Lossy data compression
  - Feature selection and
  - Data Visualization
- Principal component analysis seeks a lower dimensionality linear space (principal subspace) such that
  - The orthogonal projection of the data points onto this subspace maximizes the variance of the projected points (Hotelling, 1933)
  - An alternative definition of PCA is based on minimizing the sum-of-squares of the projection errors (Pearson, 1901)


- Jolliffe, I. T. (2002). *Principal Component Analysis* (Second ed.). Springer.
- Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* **24**, 417–441.
- Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, Sixth Series* **2**, 559–572.


# Principal Component Analysis

- PCA seeks a space of lower dimensionality (magenta line) such that:
  - (1) the orthogonal projection of the data points (red dots) onto this subspace maximizes the variance of the projected points (green dots).
  - (2) minimizing the sum-of-squares of the projection errors (blue lines)




- J. Shlens (2005). *A Tutorial on Principal Component Analysis*

# PCA: Maximum Variance Formulation

- Consider a data set of observations $\{\boldsymbol{x_n}\}, n = 1, \ldots, N$, where $\boldsymbol{x_n}$ has dimensionality $D$.

- Our goal is to project the data onto a space having dimensionality $M < D$ (principal subspace) while maximizing the variance of the projected data ($M$ here is fixed).

- Let $\{\boldsymbol{u}_i\}, i = 1, \ldots, M$ the basis vectors ($(D \times 1)$ vectors) of the principal subspace.

- We define the sample mean ($(D \times 1)$ vector) by: $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$

  and the sample covariance matrix ($(D \times D)$ matrix) by:

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)^T$$

# PCA with 1D Principal Subspace

- Consider first the projection onto 1D space ($M = 1$).
- We define the direction of this subspace using vector $\boldsymbol{u}_1$, which we choose to be a unit vector: $\boldsymbol{u}_1^T \boldsymbol{u}_1 = 1$
- Each data point $\boldsymbol{x}_n$ is then projected onto the scalar $\boldsymbol{u}_1^T \boldsymbol{x}_n$
- The mean of the projected data is given as: $\boldsymbol{u}_1^T \bar{\boldsymbol{x}}$
- Similarly, the variance of the projected data is:

$$\frac{1}{N}\sum_{n=1}^{N}\left\{\boldsymbol{u}_1^T \boldsymbol{x}_n - \boldsymbol{u}_1^T \bar{\boldsymbol{x}}\right\}^2 = \frac{1}{N}\sum_{n=1}^{N}\left\{\boldsymbol{u}_1^T \boldsymbol{x}_n - \boldsymbol{u}_1^T \bar{\boldsymbol{x}}\right\}\left\{\boldsymbol{u}_1^T \boldsymbol{x}_n - \boldsymbol{u}_1^T \bar{\boldsymbol{x}}\right\}^T = \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1$$

- Key idea of PCA: Maximize the projected variance $\boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1$ with respect to $\boldsymbol{u}_1$ under the constraint $\boldsymbol{u}_1^T \boldsymbol{u}_1 = 1$

# PCA with 1D Principal Subspace

- Introduce the Lagrange multiplier $\lambda_1$ and define the unconstrained maximization of

$$\max_{\boldsymbol{u}_1, \lambda_1} \left\{ \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 + \lambda_1 \left( 1 - \boldsymbol{u}_1^T \boldsymbol{u}_1 \right) \right\}$$

- We can see immediately that the solution satisfies:

$$\boldsymbol{S}\boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1$$

- $\boldsymbol{u}_1$ must be an eigenvector of $\boldsymbol{S}$ with eigenvalue $\lambda_1$

- From the eigen-problem note that the variance of the projected data is $\boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 = \lambda_1$ so $\lambda_1$ needs to be the largest eigenvalue of $\boldsymbol{S}$.

- $\boldsymbol{u}_1$ is called the first principal component.

# PCA with 1D Principal Subspace

- Additional Principal Components: maximize the projected variance amongst all possible directions orthogonal to those already considered.

- Using induction, you can show: For an $M$-dimensional projection space,

  The optimal linear projection for which the variance of the projected data is maximized is defined by the $M$ eigenvectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_M$ of the data covariance matrix $\boldsymbol{S}$ corresponding to the largest eigenvalues $\lambda_1, \dots, \lambda_M$.

- The computational cost of finding the $M$ principal components (i.e. finding the first $M$ eigenvalues and eigenvectors of $\boldsymbol{S}$) is $\mathcal{O}(MD^2)$.

# Total Variance

- PCA becomes the calculation of the eigenvectors of the data covariance matrix corresponding to the largest $M$ eigenvalues.
- $\sum_{i=1}^{D} \lambda_i$ is the total variance.
- The contribution to the total variance by one component $\boldsymbol{u}_i$ is measured as:

$$\frac{\lambda_i}{\sum_{i=1}^{D} \lambda_i}$$

# Proof by Induction: $M$-Dimensional Subspace

- We have shown our result for $M = 1$. Assume it is valid for projection space of dimensionality $M$. We will show that it is then valid for projection space of dimensionality $M + 1$.
- We need to maximize the following:

$$\max_{\boldsymbol{u}_{M+1}} \left\{ \boldsymbol{u}_{M+1}^T \boldsymbol{S} \boldsymbol{u}_{M+1} + \lambda_{M+1} \left(1 - \boldsymbol{u}_{M+1}^T \boldsymbol{u}_{M+1}\right) + \sum_{i=1}^{M} \eta_i \boldsymbol{u}_{M+1}^T \boldsymbol{u}_i \right\}$$

- The stationary points satisfy:

$$0 = 2\boldsymbol{S}\boldsymbol{u}_{M+1} - 2\lambda_{M+1}\boldsymbol{u}_{M+1} + \sum_{i=1}^{M} \eta_i \boldsymbol{u}_i$$

- Taking dot product with $\boldsymbol{u}_j^T$, $j = 1, \ldots, M$ gives $\eta_j = 0$. Thus

$\boldsymbol{S}\boldsymbol{u}_{M+1} = \lambda_{M+1}\boldsymbol{u}_{M+1} \Rightarrow \boldsymbol{u}_{M+1}$ eigenvector of $\boldsymbol{S}$ with the variance in this direction $\boldsymbol{u}_{M+1}^T \boldsymbol{S} \boldsymbol{u}_{M+1} = \lambda_{M+1}$

- $\lambda_{M+1}$ needs to be the next largest eigenvalue after the first $M$ ones → The argument by induction holds for any $M \leq D$.

# PCA: Minimum Error Formulation

- This is based on projection error minimization.
- Consider a $D$-dimensional basis $\{\boldsymbol{u}_i\}$ where $i = 1, \ldots, D$ satisfying $\boldsymbol{u}_1^T \boldsymbol{u}_j = \delta_{ij}$
- Each data point $\boldsymbol{x}_n$ can be represented by:

$$\boldsymbol{x}_n = \sum_{i=1}^{D} a_{ni} \boldsymbol{u}_i \,, \text{ where: } a_{ni} = \boldsymbol{x}_n^T \boldsymbol{u}_i$$

- We approximate $\boldsymbol{x}_n$ using a representation with $M < D$ variables $z_{ni}$ via a projection onto a lower-dim subspace:

$$\tilde{\boldsymbol{x}}_n = \sum_{i=1}^{M} z_{ni} \boldsymbol{u}_i + \sum_{i=M+1}^{D} b_i \boldsymbol{u}_i$$

where the $\{z_{ni}\}$ depend on the particular data point, whereas the $\{b_i\}$ are the same constants for all data points.

- We are free to choose the $\{\boldsymbol{u}_i\}, \{z_{ni}\}$, and $\{b_i\}$

# PCA: Minimum Error Formulation

- Key idea of PCA: Minimize the distortion $J$ introduced by the dimensionality reduction:

$$\min_{z_{nj},b_j,\boldsymbol{u}_i} J = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2, \quad \tilde{\boldsymbol{x}}_n = \sum_{i=1}^{M} z_{ni}\boldsymbol{u}_i + \sum_{i=M+1}^{D} b_i \boldsymbol{u}_i$$

- Setting the derivatives wrt $\{z_{ni}\}$, and $\{b_i\}$ to zero and using $\boldsymbol{u}_1^T \boldsymbol{u}_j = \delta_{ij}$ gives:

$$z_{nj} = \boldsymbol{x}_n^T \boldsymbol{u}_j, \; j = 1,\ldots,M$$

$$b_j = \bar{\boldsymbol{x}}^T \boldsymbol{u}_j, \; j = M+1,\ldots,D, \text{ where } \bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$

- If we now substitute these expressions for $\{z_{ni}\}$, and $\{b_i\}$

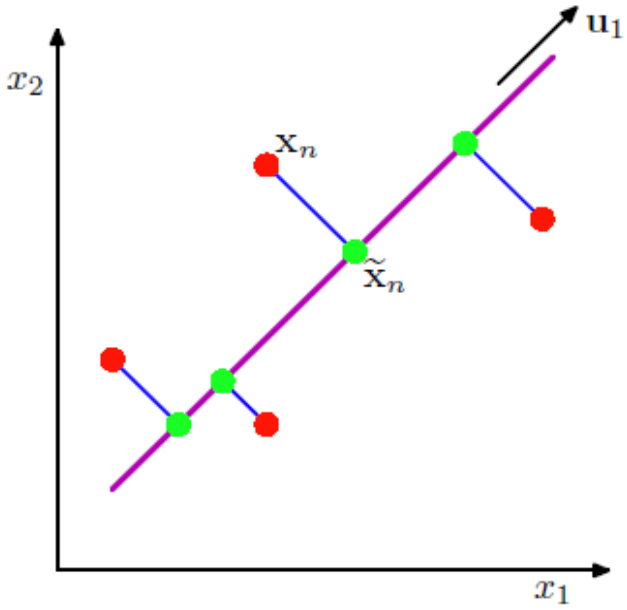
$$\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n = \sum_{i=1}^{D} (\boldsymbol{x}_n^T \boldsymbol{u}_i)\boldsymbol{u}_i - \sum_{i=1}^{M} (\boldsymbol{x}_n^T \boldsymbol{u}_i)\boldsymbol{u}_i - \sum_{i=M+1}^{D} (\bar{\boldsymbol{x}}^T \boldsymbol{u}_i)\boldsymbol{u}_i = \sum_{i=M+1}^{D} \{(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T \boldsymbol{u}_i\}\boldsymbol{u}_i$$

# PCA: Minimum Error Formulation

$$\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n = \sum_{i=M+1}^{D} \left\{ \left( \boldsymbol{x}_n - \overline{\boldsymbol{x}} \right)^T \boldsymbol{u}_i \right\} \boldsymbol{u}_i$$

- We see that the displacement vector $\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n$ lies in the space orthogonal to the principal subspace, i.e. is a linear combination of $\{\boldsymbol{u}_i\}$ for $i = M + 1, \ldots, D$.

- We can also see this geometrically:




  - The projected points $\tilde{\boldsymbol{x}}_n$ must lie within the principal subspace, but can move them freely within that subspace.

  - The minimum error is then obtained by the orthogonal projection.

# PCA: Minimum Error Formulation

- We obtain an expression for the distortion measure $J$ *as* a function purely of the $\{\boldsymbol{u}_i\}$ in the form

$$\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n = \sum_{i=M+1}^{D} \left\{ \left( \boldsymbol{x}_n - \bar{\boldsymbol{x}} \right)^T \boldsymbol{u}_i \right\} \boldsymbol{u}_i$$

$$J = \frac{1}{N} \sum_{i=1}^{N} \| \boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n \|^2$$

$$J = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} \left( \boldsymbol{x}_n^T \boldsymbol{u}_i - \bar{\boldsymbol{x}}^T \boldsymbol{u}_i \right)^2 = \sum_{i=M+1}^{D} \boldsymbol{u}_i^T \boldsymbol{S} \boldsymbol{u}_i$$

- The minimum of $J$ is obtained when $\{\boldsymbol{u}_i\}, i = M+1, \ldots, D$ are the eigenvectors of $\boldsymbol{S}$ associated to the smallest eigenvalues.

- Consider $D = 2$ and $M = 1$. Let $\lambda_1 > \lambda_2$. The principal subspace is aligned with the eigenvector with the larger eigenvalue $\lambda_1$, and the min value of $J = \lambda_2$ is obtained by choosing $\boldsymbol{u}_2$ corresponding to $\lambda_2$.

# PCA: Minimum Error Formulation

- The distortion measure is given as

$$J = \sum_{i=M+1}^{D} \lambda_i$$

- Obtain the minimum value of $J$ by selecting these eigenvectors corresponding to the $D - M$ smallest eigenvalues.

- The eigenvectors defining the principal subspace are those corresponding to the $M$ largest eigenvalues.

- The two approaches discussed
  - Maximum Variance and
  - Minimum Error

  both lead to the same main result of the PCA.

# Compression of the Original Data Set

- Using the earlier equations, we can derive:

$$\tilde{\boldsymbol{x}}_n = \sum_{i=1}^{M} z_{ni}\boldsymbol{u}_i + \sum_{i=M+1}^{D} b_i\boldsymbol{u}_i$$

$$z_{nj} = \boldsymbol{x}_n^T\boldsymbol{u}_j \;, j = 1,\ldots,M$$

$$b_j = \bar{\boldsymbol{x}}^T\boldsymbol{u}_j \;, j = M+1,\ldots,D$$

$$\Rightarrow \tilde{\boldsymbol{x}}_n = \sum_{i=1}^{M}\left(\boldsymbol{x}_n^T\boldsymbol{u}_i\right)\boldsymbol{u}_i + \sum_{i=M+1}^{D}\left(\bar{\boldsymbol{x}}^T\boldsymbol{u}_i\right)\boldsymbol{u}_i \Rightarrow$$

$$Data\ Compression: \tilde{\boldsymbol{x}}_n = \bar{\boldsymbol{x}} + \sum_{i=1}^{M}\left(\boldsymbol{x}_n^T\boldsymbol{u}_i - \bar{\boldsymbol{x}}^T\boldsymbol{u}_i\right)\boldsymbol{u}_i$$

- In deriving the last equation, we used the completeness of $\boldsymbol{u}_i$, i.e.

$$\bar{\boldsymbol{x}} = \sum_{i=1}^{D}\left(\bar{\boldsymbol{x}}^T\boldsymbol{u}_i\right)\boldsymbol{u}_i$$

- This clearly shows the compression of the data set: From the $D$-dimensional $\boldsymbol{x}_n$ to the $M$-dimensional vector with components $\left(\boldsymbol{x}_n^T\boldsymbol{u}_i - \bar{\boldsymbol{x}}^T\boldsymbol{u}_i\right)$.

# PCA Reconstruction

- To reconstruct the data in the original $D$-dimensional space from a representation in the $M$-dimensional principal subspace, we simply use:

$$\tilde{\boldsymbol{x}}_n = \overline{\boldsymbol{x}} + \sum_{i=1}^{M} \left( \boldsymbol{x}_n^T \boldsymbol{u}_i - \overline{\boldsymbol{x}}^T \boldsymbol{u}_i \right) \boldsymbol{u}_i$$

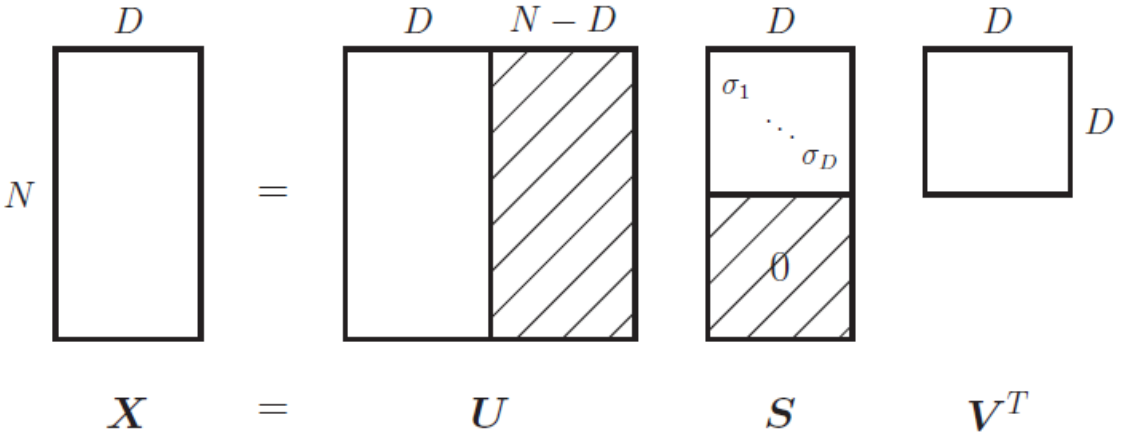
- If $M = D$, there is no dimensionality reduction but a rotation to align with the principal components e.g from $\{x_{n_1}, \dots, x_{n_D}\}$ to $\{a_{n_1}, \dots, a_{n_D}\}$ where:

$$\boldsymbol{x}_n = \sum_{i=1}^{M=D} a_{ni} \boldsymbol{u}_i$$

# PCA and Singular Value Decomposition

- We have defined the solution to PCA in terms of eigenvectors of the empirical covariance matrix (here denoted as $\hat{\boldsymbol{\Sigma}}$). The same solution can be obtained based on SVD on the standardized data matrix $\boldsymbol{X}$.

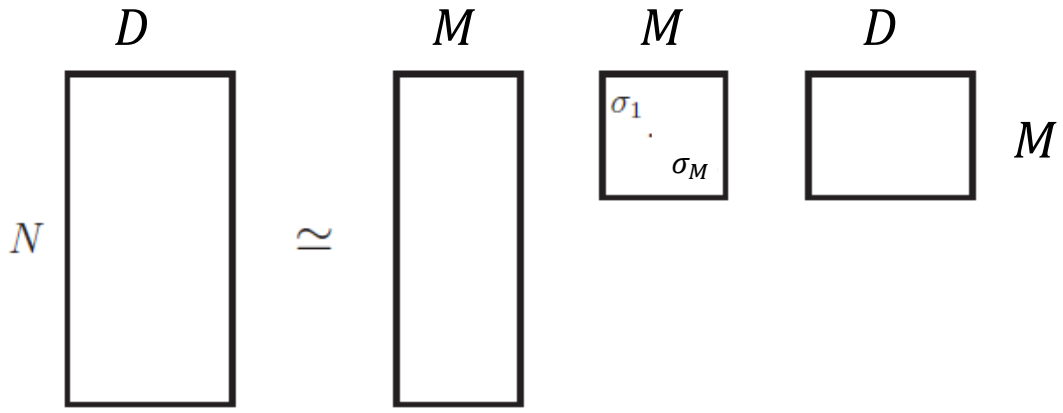
$\boldsymbol{U}, \boldsymbol{V}$: orthonormal
$\sigma_1, \sigma_2, .., \sigma_D$: singular values ($\geq 0$)




- The shaded entries in $\boldsymbol{S}$ and the off diagonal terms are zero. So we don't need to compute the shaded entries of $\boldsymbol{U}$ and $\boldsymbol{S}$.
- It can be shown that the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$ are equal to $\boldsymbol{V}$ (right singular vectors of $\boldsymbol{X}$) and the eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$ equal to $\boldsymbol{D} = \boldsymbol{S}^2$ (squared singular values) $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{S}^2\boldsymbol{V}^T \rightarrow (\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{V} = \boldsymbol{V}\boldsymbol{D}$
- Similarly: $\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T = U\boldsymbol{S}^2\boldsymbol{U}^T \rightarrow (\boldsymbol{X}\boldsymbol{X}^T)\boldsymbol{U} = \boldsymbol{U}\boldsymbol{D}$

# PCA and Singular Value Decomposition

- If the singular values die off quickly, we can produce a rank $M$ approximation of $\boldsymbol{X}$ as follows:



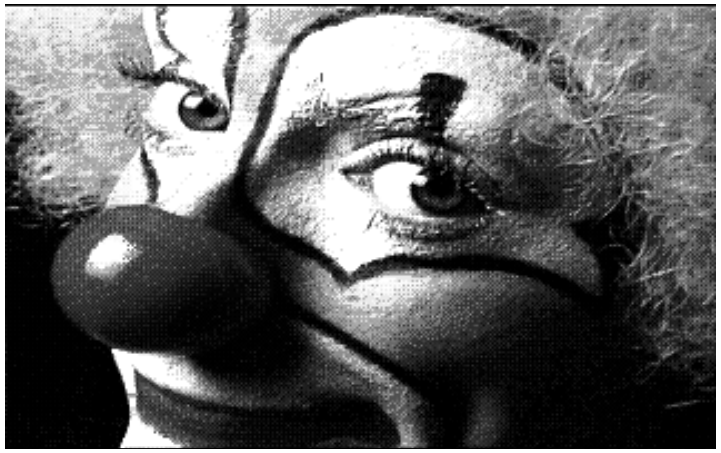

$$\boldsymbol{X} \approx \boldsymbol{U}_M \quad \boldsymbol{S}_M \quad \boldsymbol{V}_M^T$$

- The number of parameters for this truncated SVD are: $NM + M + MD$

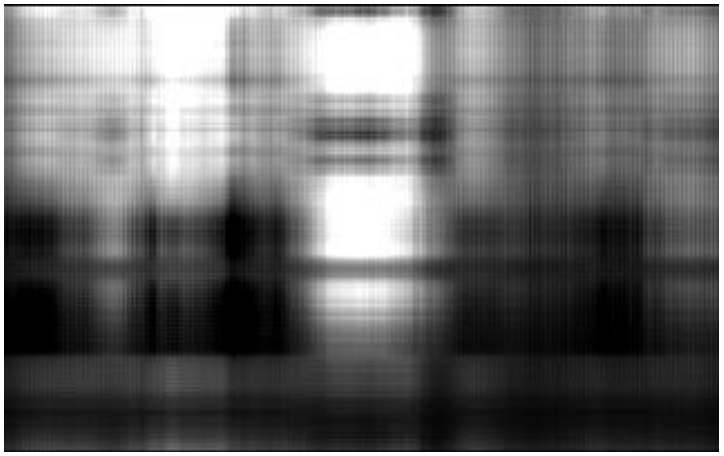
# PCA and Singular Value Decomposition

- Consider the image shown of size $200 \times 320$ (rank 200) and the SVD approximations of rank $2, 5$, and $20$.

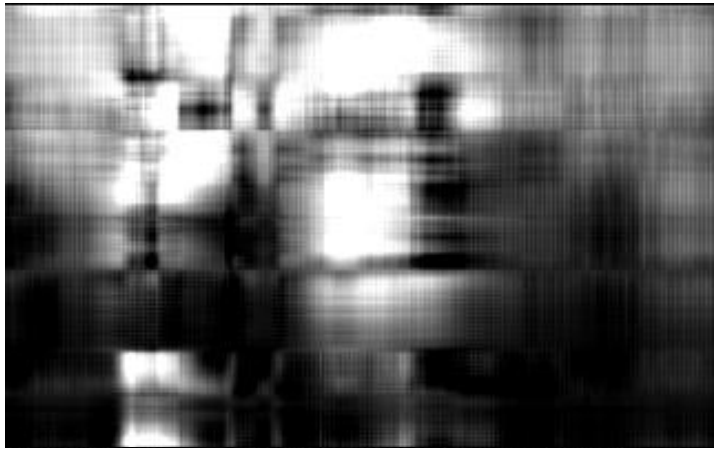
rank 200



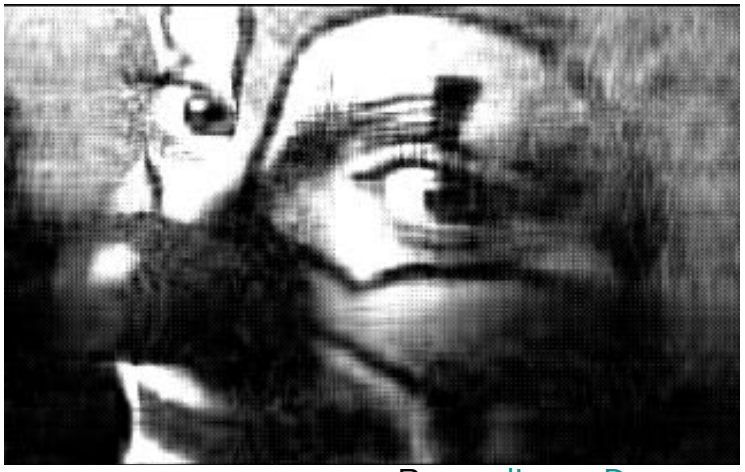
rank 2



rank 5



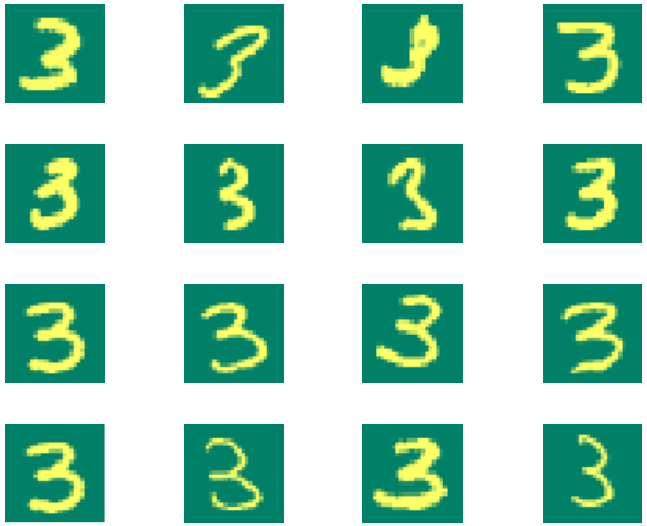
rank 20



Run svdImageDemo
From PMTK3

# Off-line Digit Images: An Example of PCA

- Let us apply PCA in the digit images discussed earlier.
- Consider a synthetic data set obtained by taking one of the digit images and creating multiple copies ($N = 10{,}000$) in each of which the digit has undergone a random displacement and rotation.
- The resulting images each have $D = 28 \times 28 = 784$ pixels.



MatLab Code

# Off-line Digit Images: PCA Algorithm

- Our dataset is $\boldsymbol{X} \in D \times N$, where $D = 784$ (pixels) and $N = 10{,}000$ (images).

- Each column of the matrix $\boldsymbol{X}$ is a data point (image) $\boldsymbol{x}_n, (n = 1, 2, \ldots, 10{,}000)$ of size $784$.

- We define the centered covariance matrix as:

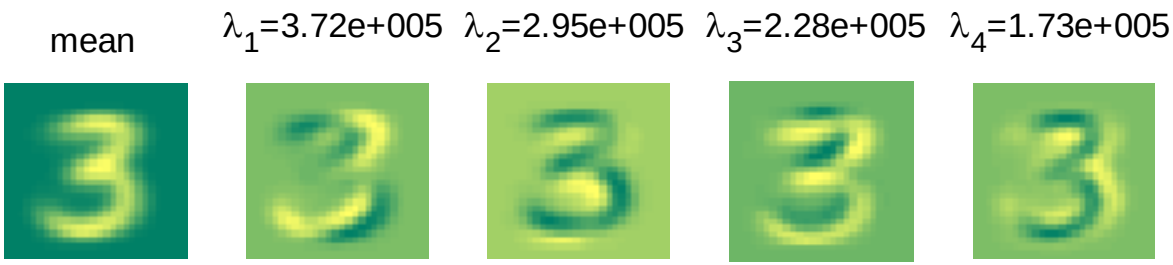
$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)^T$$

- To compute the principal components, we solve the following eigenvalue problem:

$$\boldsymbol{S}\boldsymbol{u} = \lambda\boldsymbol{u}$$

# Off-line Digit Images: PCA Results

- Using the algorithm stated before, we apply PCA on the off-line digits dataset.
- Since each eigenvector of $\boldsymbol{S}$ is a vector in the original $D$-dimensional space, we can represent the eigenvectors as images of the same size as the data points (see Fig. below)
- Below is the mean vector $\bar{\boldsymbol{x}}$ along with the first four PCA eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_4$ for the digit 3 from the off-line digits data set, together with the corresponding eigenvalues.




MatLab Code

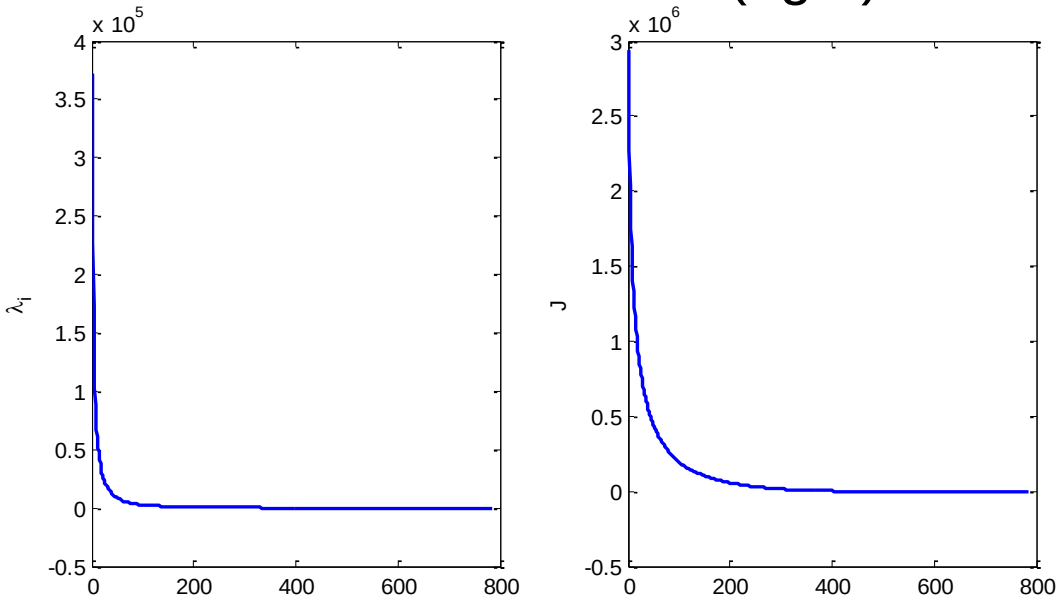
# Off-line Digit Images: PCA Results

- The spectrum of eigenvalues, sorted into decreasing order, is shown below (left).

- The distortion measure $J$ associated with choosing a particular value of $M$ is given by

$$J = \sum_{i=M+1}^{D} \lambda_i$$

and is plotted for different values of $M$ (right).




MatLab Code

# Off-line Digit Images: PCA Reconstruction

- Below shows an original example of digit 3 from the off-line digits data set together with its PCA reconstructions obtained by retaining $M$ principal components for various values of $M$.

$$\tilde{\boldsymbol{x}}_n = \overline{\boldsymbol{x}} + \sum_{i=1}^{M} \left( \boldsymbol{x}_n^T \boldsymbol{u}_i - \overline{\boldsymbol{x}}^T \boldsymbol{u}_i \right) \boldsymbol{u}_i$$

- As $M$ increases the reconstruction becomes more accurate and would become perfect when $M = D = 28 \times 28 = 784$.


MatLab Code

# Off-line Digit Images: PCA Reconstruction

mean

principal basis 1

principal basis 2

principal basis 3

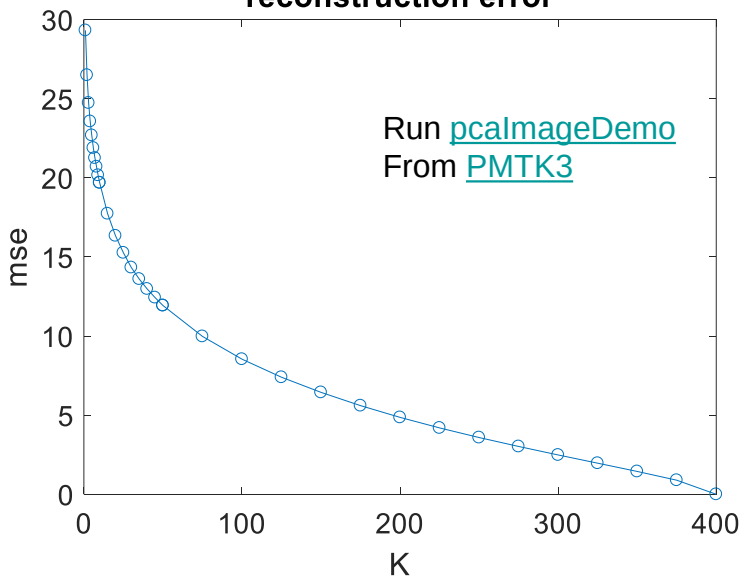
Using 5 bases

Using 10 bases

Using 20 bases

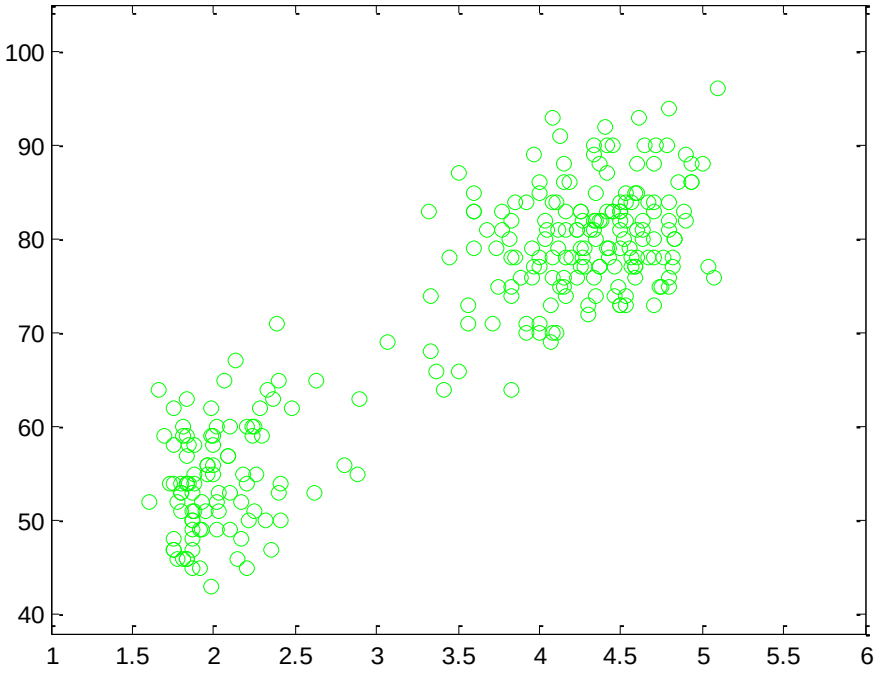
Using 400 bases

reconstruction error



# Old Faithful Data Set: Whitening

- Another application of PCA is data pre-processing (standardizing certain of its properties). This is applied here to the old faithful dataset.
- Here the goal is the transformation of a data set in order to standardize the data (zero mean/unit variance).
- The original dataset is shown below:




MatLab Code

# Old Faithful Data Set: Standardizing

- Standardizing the data is useful for successful application of pattern recognition techniques.
- It is essential when the original variables are measured in various different units or have different scaling.
- In the Old Faithful data set, the time between eruptions is typically an order of magnitude greater than the duration of an eruption.
- Before applying classification techniques (e.g. $K$ −Means) we perform linear re-scaling of the **individual variables** such that **each variable has zero mean and unit variance**.
- This is known as standardizing the data.

# Old Faithful Data Set: Standardizing

- The covariance matrix for the standardized data has components

$$\rho_{ij} = \frac{1}{N}\sum_{n=1}^{N} \frac{x_{ni} - \bar{x}_i}{\sigma_i} \frac{x_{nj} - \bar{x}_j}{\sigma_j}, \sigma_i = std\ of\ x_i$$

- This is the correlation matrix of the original data.

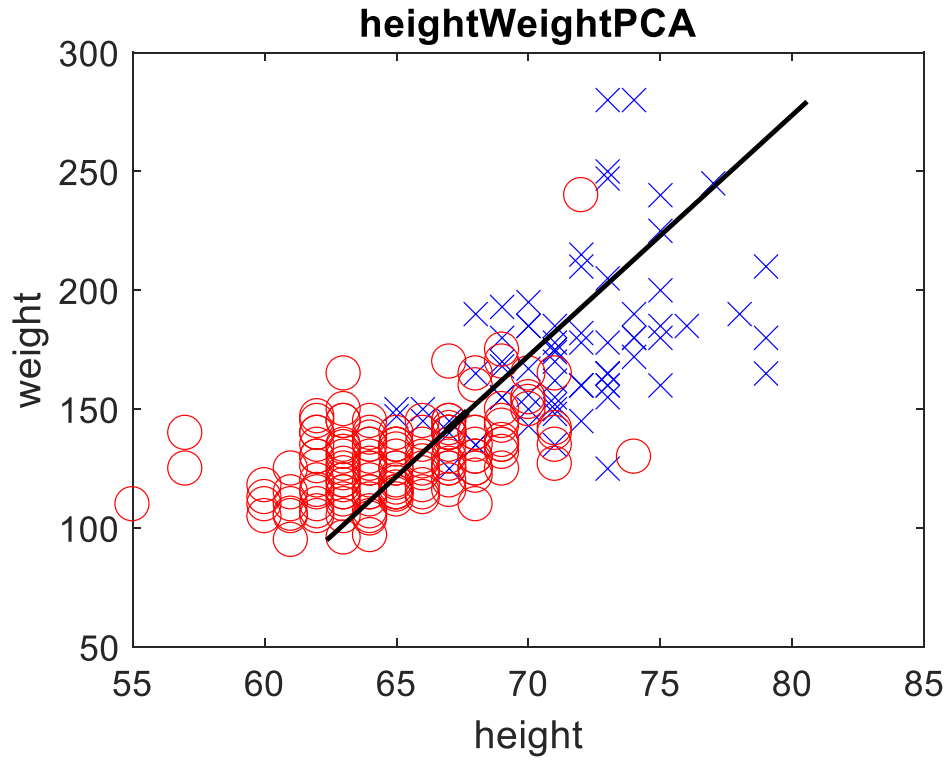
- Two components $x_i$ *and* $x_j$ of the data are

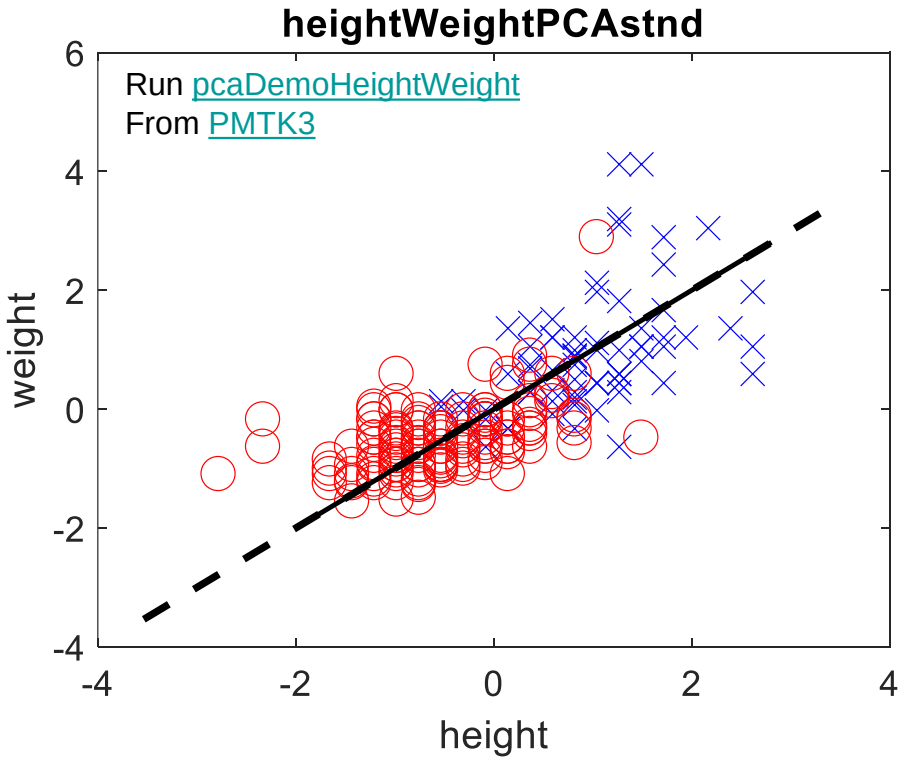
  - ✓ perfectly correlated, then $\rho_{ij} = 1$, and

  - ✓ uncorrelated, then $\rho_{ij} = 0$.

# Old Faithful Data Set: PCA Example

- It should be standard practice to standardize the data first before performing PCA. This is equivalent to working with correlation matrices instead of covariance matrices.
- (Left) PCA for raw data (Right) PCA of standardized data (individual variables having zero mean and unit variance).



# Old Faithful Data Set: Whitening

- With PCA we can normalize the data to give them zero mean and **unit covariance** (**different variables become decorrelated**).

- Consider the key eigenvalue problem in PCA in a matrix form:

$$\boldsymbol{SU} = \boldsymbol{UL}, \quad \boldsymbol{L} = diag\left(\lambda_1, ..., \lambda_D\right), \quad \boldsymbol{U} = \left[\boldsymbol{u}_1 ... \boldsymbol{u}_D\right] (orthogonal)$$

- For each data point $\boldsymbol{x}_n$, define a transformed value as:

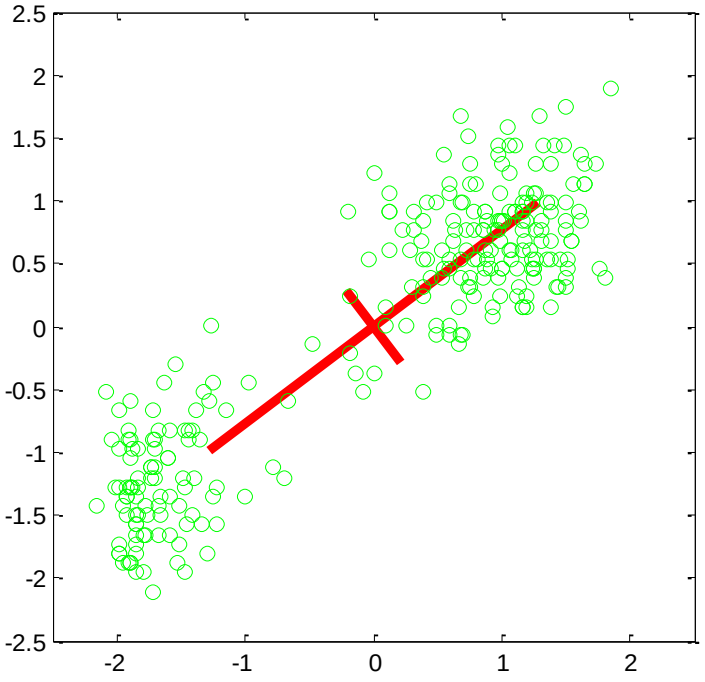
$$\text{Whitening of the data}: \boldsymbol{y}_n = \boldsymbol{L}^{-1/2}\boldsymbol{U}^T\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)$$

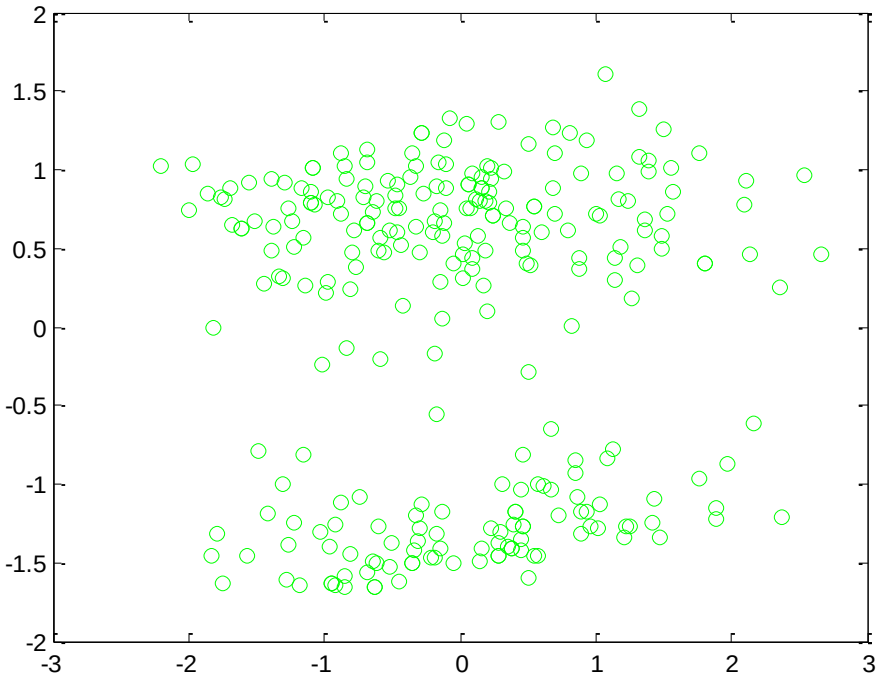
- The set $\{\boldsymbol{y}_n\}$ has zero mean and its covariance is the identity:

$$\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{y}_n\boldsymbol{y}_n^T = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{L}^{-1/2}\boldsymbol{U}^T\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)^T\boldsymbol{U}\boldsymbol{L}^{-1/2} = \boldsymbol{L}^{-1/2}\boldsymbol{U}^T\boldsymbol{S}\boldsymbol{U}\boldsymbol{L}^{-1/2} = \boldsymbol{I}$$

# Old Faithful Data Set: PCA Example

- Left: Standardizing individual variables to zero mean and unit variance. The principal axes of the normalized set are shown for the range $\pm\lambda_i^{1/2}$ (variables still correlated)
- Right: Whitening of the data (zero mean, unit covariance)

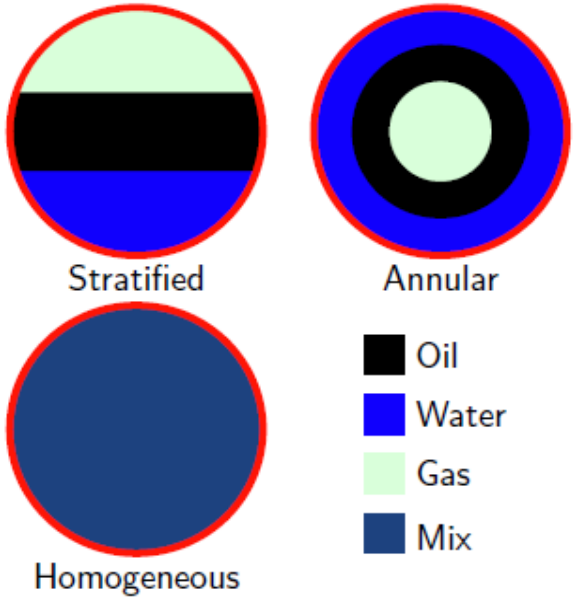



MatLab Code

# Oil Flow Data Set

- Consider the oil flow data set.
- For a given geometrical configuration of the gas, water, and oil phases) there are only 2 DOF of variability
  - ✓ the fraction of oil in the pipe and
  - ✓ the fraction of water (the fraction of gas then being determined).
- The data space comprises 12 measurements.
- However, the points in the data set lie close to a 2D manifold embedded within this $12^{th}$ dimensional space.

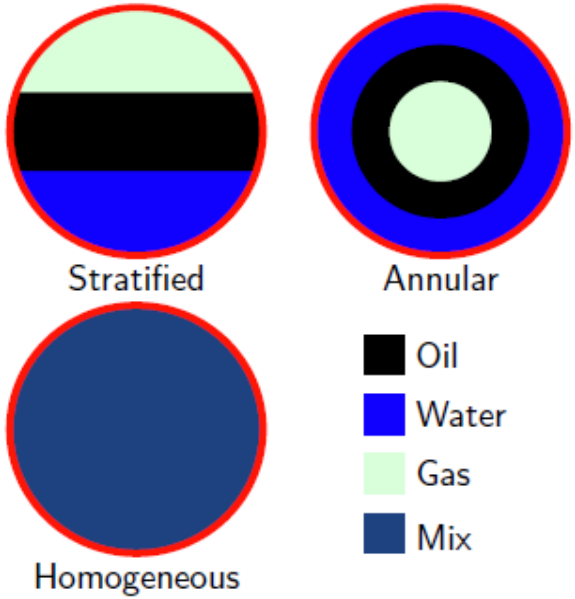



**Flow Configurations**

Red : 'laminar'
Blue: 'homogeneous'
Green: 'annular'

# Oil Flow Data Set

- The manifold comprises several distinct segments corresponding to the different flow regimes, each such segment being a (noisy) continuous 2D manifold.
- For data compression, there is merit in exploiting this manifold structure.

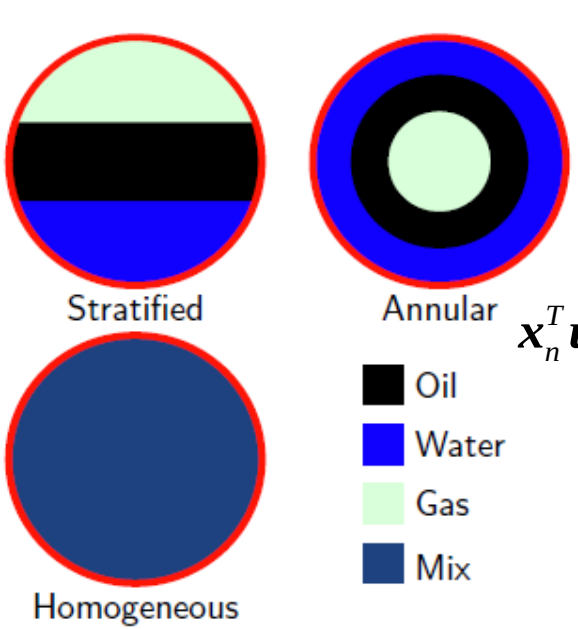


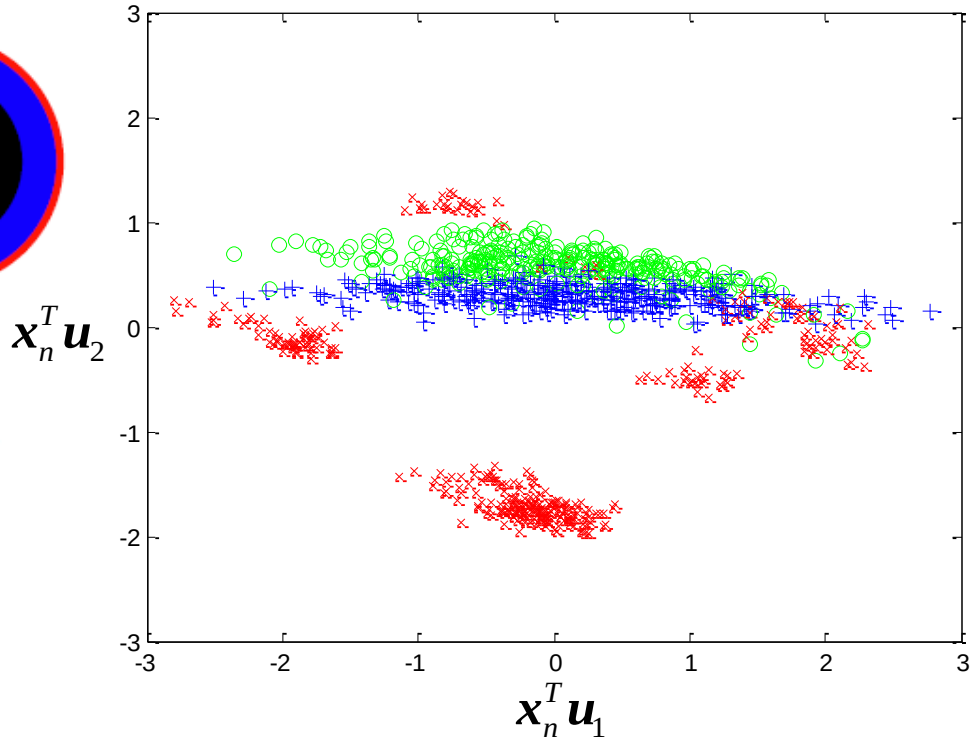

**Flow Configurations**

Red : 'laminar'
Blue: 'homogeneous'
Green: 'annular'

# Oil Flow Data Set: PCA for Visualization

- Visualization of the oil flow data set obtained by projecting the data onto the first two principal components.



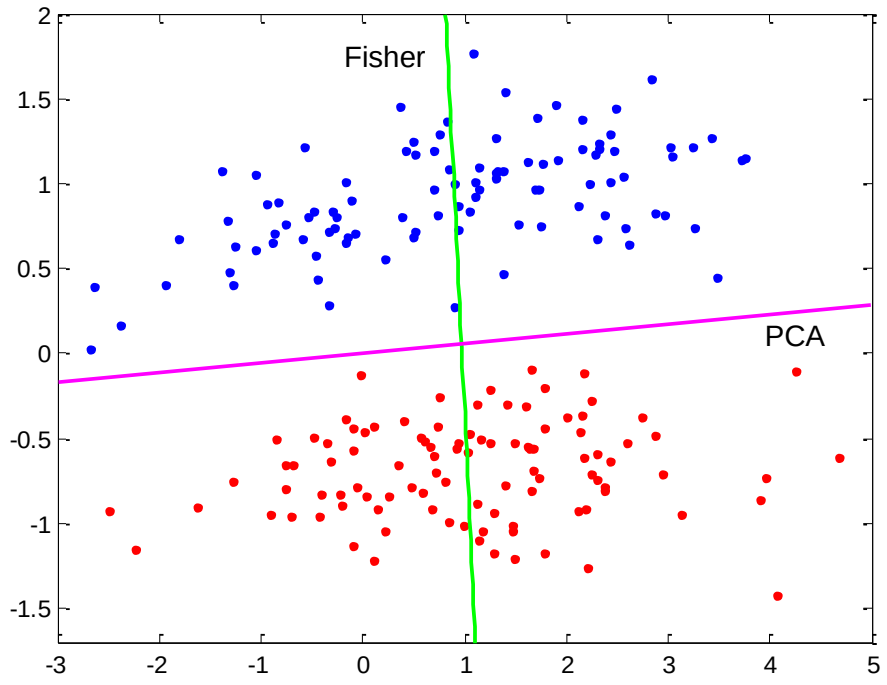




**Flow Configurations**
Red : 'laminar'
Blue: 'homogeneous'
Green: 'annular'

MatLab Code

# PCA Vs. Fisher's Discriminant Method

- PCA and the Fisher linear discriminant method can be seen as linear dimensionality reduction techniques.
- PCA is unsupervised and depends only on $\boldsymbol{x_n}$ whereas Fisher linear discriminant also uses class-label information.




MatLab Code

- 2D data, 2 classes
- We are interested projecting the data in 1D.
- PCA chooses the direction of maximum variance (magenta curve) leading to to strong class overlap.
- The Fisher linear discriminant projects (green curve) to maximize class separation.

# PCA For High Dimensional Data

- In many applications of PCA, the number of data points ($N$) is smaller than the dimensionality ($D$) of the data space (e.g. $100$ images each with $100{,}000$ pixels).

- $N$ points in a $D$ −dimensional space, where $N << D$, defines a linear subspace whose dimensionality is at most $N - 1$.

  - There is no point in applying PCA for values of $M$ that are greater than $N - 1$.

  - If we perform PCA, we will find that at least $D - N + 1$ of the eigenvalues are zero, corresponding to eigenvectors along whose directions the data set has zero variance.

# PCA For High Dimensional Data

- Typical algorithms for finding the eigenvectors of a $D \times D$ matrix have a computational cost that scales like $\mathcal{O}(D^3)$.

- Direct application of PCA in e.g. the image example will be computationally infeasible.

# PCA For High Dimensional Data

- To resolve the problem, let $\boldsymbol{X}$ to be the $N \times D$-dimensional centered data matrix, whose $n^{\text{th}}$ row is given by $\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)^T$ .

- The covariance $D \times D$ matrix can then be written as:

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)^T = \frac{1}{N}\boldsymbol{X}^T\boldsymbol{X}$$

- The corresponding eigenvector equation is:

$$\frac{1}{N}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

- Pre-multipling both sides from the left by $\boldsymbol{X}$ gives:

$$\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T\left(\boldsymbol{X}\boldsymbol{u}_i\right) = \lambda_i\left(\boldsymbol{X}\boldsymbol{u}_i\right)$$

# PCA For High Dimensional Data

$$\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T\left(\boldsymbol{X}\boldsymbol{u}_i\right)=\lambda_i\left(\boldsymbol{X}\boldsymbol{u}_i\right)$$

- Let us define:

$$\boldsymbol{v}_i=\boldsymbol{X}\boldsymbol{u}_i$$

- The eigenvalue problem becomes:

$$\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{v}_i=\lambda_i\boldsymbol{v}_i$$

- This is an eigenvector equation for the $N \times N$ matrix $\boldsymbol{X}\boldsymbol{X}^T/N$
- This has the same $N-1$ eigenvalues as the original covariance matrix (which has an additional $D-N+1$ zero eigenvalues). Note we have $N-1$ (and not $N$) eigenvalues because the data are centered ($\boldsymbol{X}\boldsymbol{X}^T$ has rank $N-1$).
- Thus we can solve the eigenvector problem in spaces of lower dimensionality with computational cost $\mathcal{O}(N^3)$.

# PCA For High Dimensional Data

- To determine the eigenvectors $\boldsymbol{u}_i$, we multiply by $\boldsymbol{X}^T$: .

$$\frac{1}{N}\left(\boldsymbol{X}^T\boldsymbol{X}\right)\boldsymbol{X}^T\boldsymbol{v}_i = \lambda_i\left(\boldsymbol{X}^T\boldsymbol{v}_i\right)$$

- The covariance matrix recall has eigenvectors $\boldsymbol{u}_i$. Thus with proper rescaling (assuming $\boldsymbol{v}_i$ is already normalized):

$$\boldsymbol{u}_i = \frac{1}{\left(N\lambda_i\right)^{1/2}}\boldsymbol{X}^T\boldsymbol{v}_i$$

- This approach is indeed simple:
  - first evaluate $\boldsymbol{X}\boldsymbol{X}^T$ and then find its eigenvectors and eigenvalues and
  - then compute the eigenvectors in the original data space from the equ. above.

# Probabistic PCA: Maximum Likelihood and EM Algorithm

# Probabilistic PCA

PCA as the maximum likelihood solution of a probabilistic latent variable model (Tipping & Bishop 1997, 1999, Roweis, 1998)

- Constrained form of the Gaussian distribution: the number of free parameters is restricted while the model still captures dominant correlations in the data.
- Derive an efficient EM algorithm for PCA.
- Allows dealing with missing values in the dataset.
- Mixture of probabilistic PCA models.


- Tipping, M. E. and C. M. Bishop (1997). Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University.
- Tipping, M. E. and C. M. Bishop (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B* **21**(3), 611–622.
- Roweis, S. (1998). EM algorithms for PCA and SPCA. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 626–632. MIT Press.

# Probabilistic PCA

PCA as the maximum likelihood solution of a probabilistic latent variable model (Tipping & Bishop 1997, 1999, Roweis, 1998)

- Bayesian treatment of PCA -- the dimensionality of the principal subspace can be found automatically.
- Probabilistic PCA can model class conditional densities and thus be used for classification.
- Can be used generatively: sample from the distribution.


- Tipping, M. E. and C. M. Bishop (1999a). Mixtures of probabilistic principal component analyzers. *Neural Computation* **11**(2), 443–482.


# Probabilistic PCA – Model

- Introduce a latent variable $\boldsymbol{z}$ corresponding to the principal-component subspace.
- Define a Gaussian prior distribution $p(\boldsymbol{z})$, together with a Gaussian conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ for the observed variable $\boldsymbol{x}$ conditioned on the value of $\boldsymbol{z}$.
- The prior distribution over $\boldsymbol{z}$ is given by

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$$

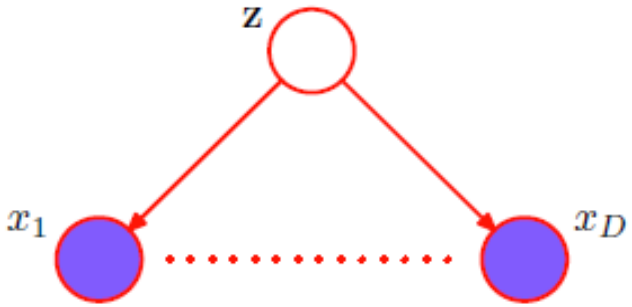
- The conditional distribution of the observed variable $\boldsymbol{x}$, conditioned on the value of $\boldsymbol{z}$, is again Gaussian:

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$$

The mean of $\boldsymbol{x}$ is a linear function of $\boldsymbol{z}$ governed by the $D \times M$ matrix $\boldsymbol{W}$ and the $D$ –dimensional vector $\boldsymbol{\mu}$.

# Probabilistic PCA: Naïve Bayes Model

- This model $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$ is an example of Naive Bayes' model - Conditioned on $\boldsymbol{z}$, the components of the observed vector $\boldsymbol{x} = (x_1, \ldots, x_D)^T$ are assumed to be independent.




- We have 2 parameters in the model:
  - (a) $\boldsymbol{W}$, its columns span a linear subspace within the data space that corresponds to the principal subspace;
  - (b) $\sigma^2$ – variance of the conditional distribution.

- The prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$ is without loss of generality.

# Probabilistic PCA – Generative View Point

- A latent variable model seeks to relate a $D$-dimensional observation vector $\boldsymbol{x}$ to a corresponding $M$-dimensional Gaussian latent variable $\boldsymbol{z}$

$$\boldsymbol{x} = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon}$$

where

- $\boldsymbol{z}$ is an $M$ −dimensional Gaussian latent variable
- $\boldsymbol{W}$ is an $(D \times M)$ matrix (the latent space)
- $\boldsymbol{\varepsilon}$ is a $D$ −dimensional Gaussian noise
- $\boldsymbol{\varepsilon}$ and $\boldsymbol{z}$ are independent
- $\boldsymbol{\mu}$ is a parameter vector (non zero mean)

- Factor analysis:

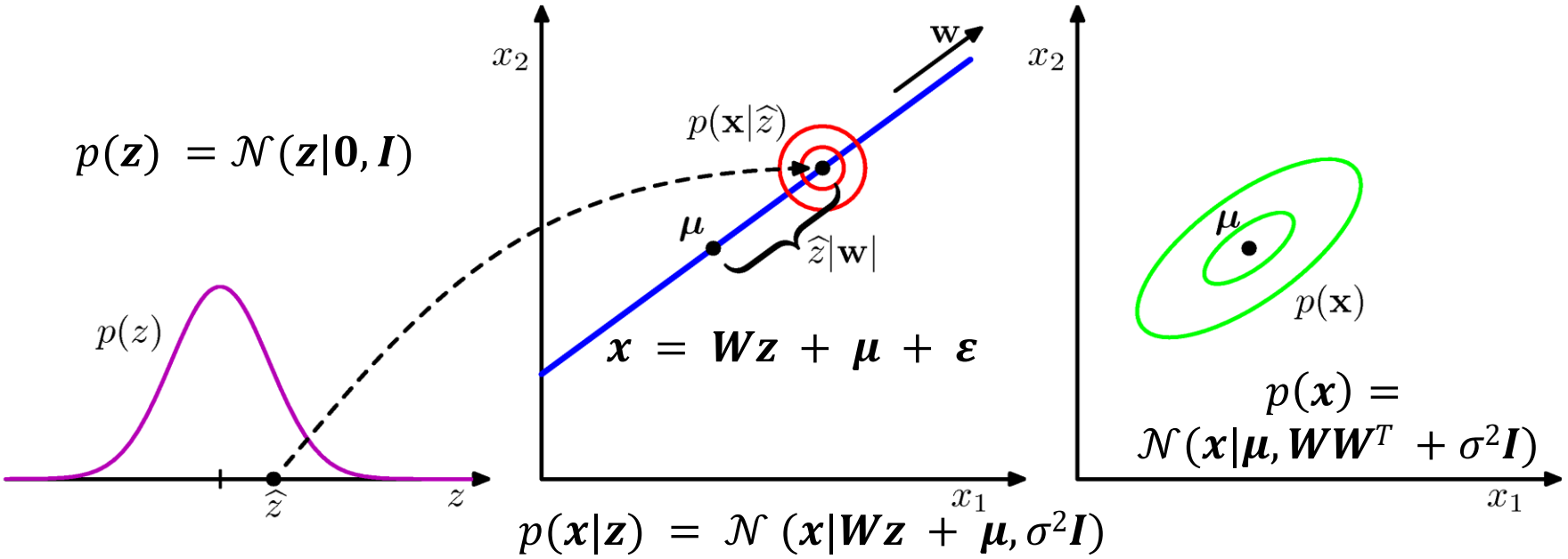
$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\psi})$$

- Probabilistic PCA:

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$$

# Probabilistic PCA – Generative View Point

- Mapping from the latent space to the data space.
- Assume here 2D data and 1D latent space.
- An observed $\boldsymbol{x}$ is generated by drawing a value $\hat{z}$ from $p(\boldsymbol{z})$ & then a value for $\boldsymbol{x}$ from an isotropic Gaussian distribution (red circles) having mean $\boldsymbol{w}\hat{z} + \boldsymbol{\mu}$ and covariance $\sigma^2\boldsymbol{I}$. The green ellipses are the density contours of $p(\boldsymbol{x})$.



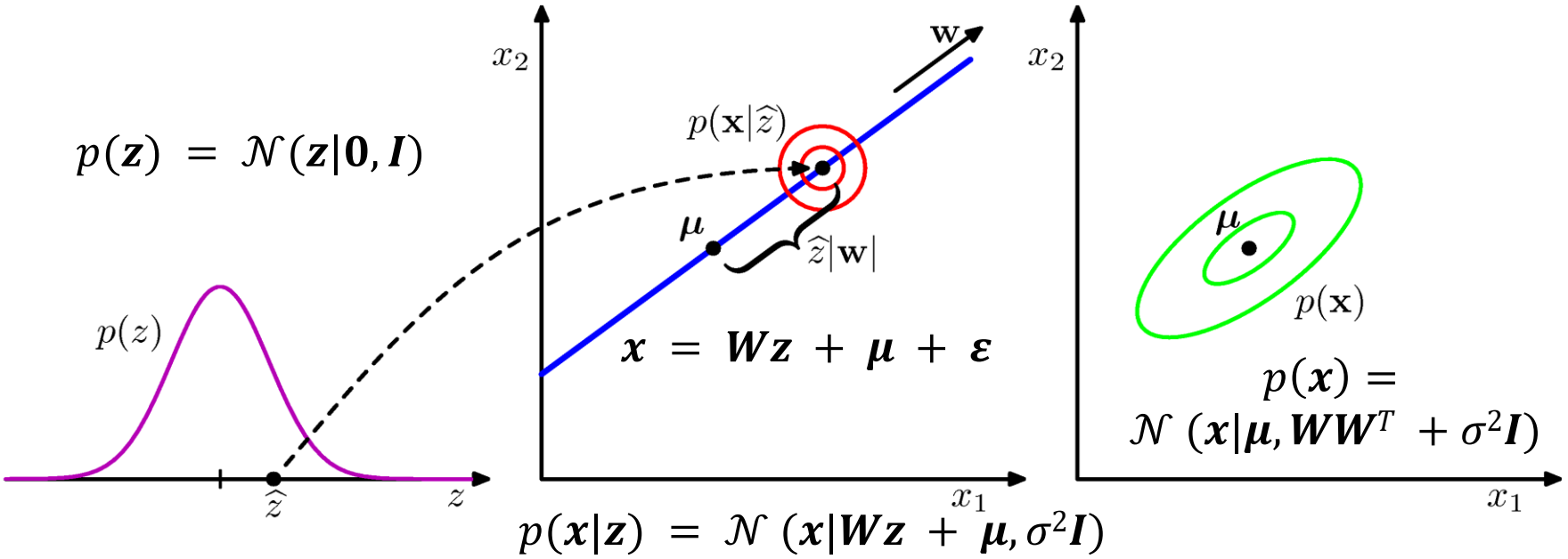
# Probabilistic PCA – Generative View Point

- To compute the likelihood function, we need an expression for the marginal distribution $p(\boldsymbol{x})$ of the observed variable:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{z})\, p(\boldsymbol{z}) d\boldsymbol{z}$$

- Using Eqs. for linear Gaussian models, this is given as:

$$p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^T + \sigma^2 \boldsymbol{I}\right)$$



# Probabilistic PCA – Generative View Point

$$p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^T + \sigma^2 \boldsymbol{I}\right)$$

- This result can be derived directly by noting that the predictive distribution will be Gaussian and then evaluating its mean and covariance.

$$\mathbb{E}[\boldsymbol{x}] = \mathbb{E}[\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon}] = \boldsymbol{W}\mathbb{E}[\boldsymbol{z}] + \boldsymbol{\mu} + \mathbb{E}[\boldsymbol{\varepsilon}] = \boldsymbol{\mu}$$

$$\text{var}[\boldsymbol{x}] = \mathbb{E}\left[(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon})(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon})^T\right]$$

$$= \boldsymbol{W}\text{var}[\boldsymbol{z}]\boldsymbol{W}^T + \mathbb{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T\right] = \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}$$

- We used here that $\boldsymbol{z}$ and $\boldsymbol{\varepsilon}$ are independent, thus uncorrelated.

# The Assumed Prior $p(z)$ is Not Restrictive

- The prior we assumed is not restrictive. Indeed consider a more general prior of the form:

$$p(\mathbf{z}) = \mathcal{N}\left(\mathbf{z} \mid \mathbf{m}, \boldsymbol{\Sigma}\right)$$

- Then the marginal distribution is of the form:

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \mid \mathbf{W}\mathbf{m} + \boldsymbol{\mu}, \mathbf{W}\boldsymbol{\Sigma}^{-1}\mathbf{W}^{T} + \sigma^{2}\mathbf{I}\right)$$

- Thus the marginal distribution has an identical form as before:

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \mid \tilde{\boldsymbol{\mu}}, \widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^{T} + \sigma^{2}\mathbf{I}\right), \qquad \tilde{\boldsymbol{\mu}} = \mathbf{W}\mathbf{m} + \boldsymbol{\mu}, \quad \widetilde{\mathbf{W}} = \boldsymbol{\Sigma}^{-1/2}\mathbf{W}$$

- We choose the simplest prior: $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})$

# Probabilistic PCA – Generative View Point

- There is redundancy in this parametrization.
- There is a whole class of $\boldsymbol{W}$'s differing by a rotation of the latent space coordinates that leads to the same predictive distribution. Indeed:
  - Let

$$\widetilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{R}, \; where \; \boldsymbol{R} = orthogonal\, matrix$$

  - Then

$$\widetilde{\boldsymbol{W}}\widetilde{\boldsymbol{W}}^T = \boldsymbol{W}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^T = \boldsymbol{W}\boldsymbol{W}^T$$

# Probabilistic PCA – Predictive Distribution

$$p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{C}\right) = \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^T + \sigma^2 \boldsymbol{I}\right)$$

- To compute the predictive distribution, we need to be able to invert $\boldsymbol{C}$ ($D \times D$ matrix). We use the matrix inversion Lemma:

$$\boldsymbol{C}^{-1} = \left(\boldsymbol{W}\boldsymbol{W}^T + \sigma^2 \boldsymbol{I}\right)^{-1} = \sigma^{-2}\boldsymbol{I} - \sigma^{-2}\boldsymbol{W}\boldsymbol{M}^{-1}\boldsymbol{W}^T$$

where

$$\boldsymbol{M} = \boldsymbol{W}^T\boldsymbol{W} + \sigma^2 \boldsymbol{I} \; (M \times M \; matrix)$$

- The cost of this inversion is reduced from $\mathcal{O}(D^3)$ to $\mathcal{O}(M^3)$!

# Probabilistic PCA – Predictive Distribution

$$\boldsymbol{C}^{-1} = \left(\boldsymbol{W}\boldsymbol{W}^T + \sigma^2 \boldsymbol{I}\right)^{-1} = \sigma^{-2}\boldsymbol{I} - \sigma^{-2}\boldsymbol{W}\boldsymbol{M}^{-1}\boldsymbol{W}^T$$

$$\boldsymbol{M} = \boldsymbol{W}^T\boldsymbol{W} + \sigma^2 \boldsymbol{I} \; (M \times M \; matrix)$$

- To derive this we used the Woodbury identity:

$$\left(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}\right)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}$$

- This identity (easy to show with direct substitution) is used often when
  - $\boldsymbol{A}$ is large and diagonal (so easy to invert),
  - $\boldsymbol{B}$ has many rows but few columns $\boldsymbol{B}$ (and conversely for $\boldsymbol{C}$ $\boldsymbol{C}$) so that the rhs is much cheaper to evaluate than the lhs.

# Probabilistic PCA – Posterior Distribution

- The posterior distribution can be derived directly from earlier results on linear Gaussian models.
  - We know $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$
  - The conditional: $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$
  - From these we conclude:

$$p(\boldsymbol{z}\,|\,\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}\,|\,\boldsymbol{M}^{-1}\boldsymbol{W}^T(\boldsymbol{x}-\boldsymbol{\mu}), \sigma^2\boldsymbol{M}^{-1}\right), \quad \boldsymbol{M} = \boldsymbol{W}^T\boldsymbol{W} + \sigma^2\boldsymbol{I}$$

Here we used results for linear Gaussian models:

$$\begin{cases} p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}\,|\,\boldsymbol{\mu}, \Lambda^{-1}\right) \\ p(\boldsymbol{y}\,|\,\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{y}\,|\,\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}, \boldsymbol{L}^{-1}\right) \end{cases} \Rightarrow$$

$$p(\boldsymbol{x}\,|\,\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{x}\,|\,\boldsymbol{\Sigma}\left(\Lambda\boldsymbol{\mu} + \boldsymbol{A}^T\boldsymbol{L}(\boldsymbol{y}-\boldsymbol{b})\right), \boldsymbol{\Sigma}\right),$$

$$\boldsymbol{\Sigma} = \left(\Lambda + \boldsymbol{A}^T\boldsymbol{L}\boldsymbol{A}\right)^{-1}$$

# Maximum Likelihood PCA

- Consider determining the model parameters using maximum likelihood. Using

$$p(\boldsymbol{x}) = \mathcal{N}\left( \boldsymbol{x} \mid \boldsymbol{\mu}, \underbrace{\boldsymbol{W}\boldsymbol{W}^T + \sigma^2 \boldsymbol{I}}_{\boldsymbol{C}} \right)$$

we derive:

$$\ln p\left( \boldsymbol{X} \mid \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2 \right) = \sum_{n=1}^{N} \ln p\left( \boldsymbol{x}_n \mid \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2 \right)$$

$$= -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{C}| - \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})^T \boldsymbol{C}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu})$$

- Setting the derivative wrt $\boldsymbol{\mu}$ equal to zero gives:

$$\boldsymbol{\mu} = \sum_{n=1}^{N} \boldsymbol{x}_n \Big/ N \equiv \overline{\boldsymbol{x}}$$

# Maximum Likelihood PCA

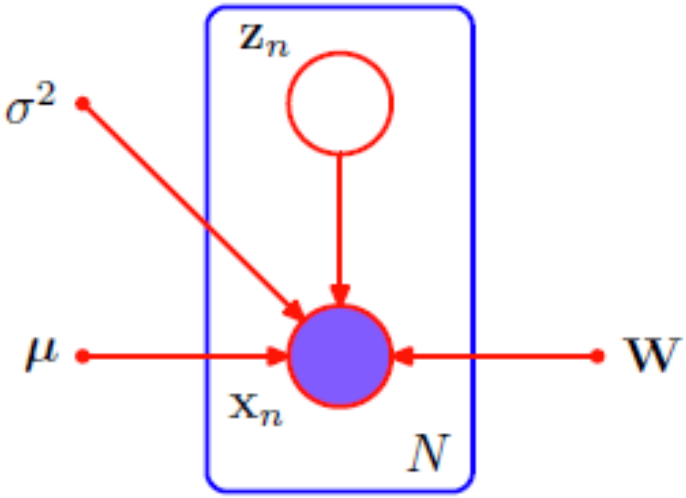
- The log-likelihood is then simplified as:

$$\ln p\left(\boldsymbol{X} \mid \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2\right) = -\frac{ND}{2}\ln\left(2\pi\right) - \frac{N}{2}\ln\left|\boldsymbol{C}\right| - \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T \boldsymbol{C}^{-1}(\boldsymbol{x}_n - \bar{\boldsymbol{x}})$$

or

$$\ln p\left(\boldsymbol{X} \mid \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2\right) = -\frac{N}{2}\left\{D\ln\left(2\pi\right) + \ln\left|\boldsymbol{C}\right| + Tr\left(\boldsymbol{C}^{-1}\boldsymbol{S}\right)\right\},$$

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T$$




- Maximization wrt $\boldsymbol{W}$ and $\sigma^2$ can also be done analytically:

$$\boldsymbol{W}_{ML} = \boldsymbol{U}_M\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2}\boldsymbol{R}$$


- Tipping, M. E. and C. M. Bishop (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B* **21**(3), 611–622.
- Roweis, S. (1998). EM algorithms for PCA and SPCA. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 626–632. MIT Press.

# Maximum Likelihood PCA

$$\boldsymbol{W}_{ML} = \boldsymbol{U}_M \left( \boldsymbol{L}_M - \sigma^2 \boldsymbol{I} \right)^{1/2} \boldsymbol{R}$$

- $\boldsymbol{U}_M$ is a $D \times M$ matrix whose columns are given by any subset (of size $M$) of the eigenvectors of the data covariance matrix $\boldsymbol{S}$.
- $\boldsymbol{L}_M$ is the $M \times M$ diagonal matrix with elements given by the corresponding eigenvalues $\lambda_i$ of $\boldsymbol{S}$.
- $\boldsymbol{R}$ is an arbitrary $M \times M$ orthogonal matrix.
- The max of the likelihood function is obtained when the $M$ eigenvectors are chosen to be those whose eigenvalues are the $M$ largest (all other solutions being saddle points).
- For eigenvectors arranged in order of decreasing $\lambda_i$ values, the $M$ principal eigenvectors are $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M$. The columns of $\boldsymbol{W}$ then define the principal subspace as in standard PCA.


- Tipping, M. E. and C. M. Bishop (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B* **21**(3), 611–622.

# Maximum Likelihood PCA

- The corresponding MLE solution for $\sigma^2$ is given as:

$$\sigma^2_{ML} = \frac{1}{D-M}\sum_{i=M+1}^{N}\lambda_i$$

  i.e. the average of the discarded eigenvalues.

- $\boldsymbol{R}$ in $\boldsymbol{W}_{ML} = \boldsymbol{U}_M\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2}\boldsymbol{R}$ is a rotation matrix in the $M$ dimensional latent space.

- Substituting this into the predictive variance $\boldsymbol{C} = \boldsymbol{W}_{ML}\boldsymbol{W}_{ML}^T + \sigma^2_{ML}\boldsymbol{I}$ and using $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$, we see that $\boldsymbol{C}$ is independent of $\boldsymbol{R}$.

- The predictive density is unchanged by rotations in the latent space (statistical non-identifiability).

# Maximum Likelihood PCA

$$\boldsymbol{W}_{ML} = \boldsymbol{U}_M \left( \boldsymbol{L}_M - \sigma^2 \boldsymbol{I} \right)^{1/2} \boldsymbol{R}$$

- For $\boldsymbol{R} = \boldsymbol{I}$, the columns of $\boldsymbol{W}$ are the principal component eigenvectors scaled by $(\lambda_i - \sigma^2)^{1/2}$.

- Interpretation: for the convolution of independent Gaussian distributions, here of $\boldsymbol{z}$ and the noise $\boldsymbol{\varepsilon}$, the variances are additive.

$$\boldsymbol{x} = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon}$$

- The variance $\lambda_i$ in the direction of an eigenvector $\boldsymbol{u}_i$ is composed of the sum of a contribution $\lambda_i - \sigma^2$ from the projection of the unit-variance latent space $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$ into data space through the corresponding column of $\boldsymbol{W}$, plus an isotropic contribution of variance $\sigma^2$ added in all directions by the noise model $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$.

# Maximum Likelihood PCA

- Consider the variance of the predictive distribution $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{m}, \boldsymbol{C})$, $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}$ along some direction defined by the unit vector $\boldsymbol{v}$ given by $\boldsymbol{v}^T\boldsymbol{C}\boldsymbol{v}$.

- Let $\boldsymbol{v}$ be orthogonal to the principal subspace $\boldsymbol{U}$ , i.e. $\boldsymbol{v}$ is given by some linear combination of the discarded eigenvectors.

- Then $\boldsymbol{v}^T\boldsymbol{U} = 0$ and hence

$$\boldsymbol{v}^T\boldsymbol{C}\boldsymbol{v} = \boldsymbol{v}^T\left(\boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}\right)\boldsymbol{v} = \sigma^2$$

- Thus the model predicts a noise variance orthogonal to the principal subspace which it was shown to be the average of the discarded eigenvalues.

$$\sigma^2_{ML} = \frac{1}{D-M}\sum_{i=M+1}^{N}\lambda_i$$

# Maximum Likelihood PCA

- Consider the variance of the predictive distribution $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{m}, \boldsymbol{C})$, $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}$ along some direction defined by the unit vector $\boldsymbol{v}$ given by $\boldsymbol{v}^T\boldsymbol{C}\boldsymbol{v}$.

- Let now $\boldsymbol{v} = \boldsymbol{u}_i$ where $\boldsymbol{u}_i$ is one of the retained eigenvectors defining the principal subspace.

- Then using $\boldsymbol{W}_{ML} = \boldsymbol{U}_M \left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2} \boldsymbol{R}$, and $\boldsymbol{u}_i^T\boldsymbol{u}_j = 0, j = 1, \ldots, M$ we see that:

$$\boldsymbol{v}^T\boldsymbol{C}\boldsymbol{v} = \boldsymbol{u}_i^T\left(\boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}\right)\boldsymbol{u}_i = \left(\lambda_i - \sigma^2\right) + \sigma^2 = \lambda_i$$

  i.e. the model correctly captures the variance of the data along the principal axes.

- As shown earlier, the variance in all remaining directions is approximated with a single value of $\sigma^2$. Variance is `lost' in the projections.

# Maximum Likelihood PCA

$$\boldsymbol{W}_{ML} = \boldsymbol{U}_M \left( \boldsymbol{L}_M - \sigma^2 \boldsymbol{I} \right)^{1/2} \boldsymbol{R}$$

$$\sigma^2_{ML} = \frac{1}{D-M} \sum_{i=M+1}^{N} \lambda_i$$

- We construct the MLE model by solving the underlying eigenvalue problem for the data covariance matrix and evaluate $\boldsymbol{W}$ and $\sigma^2$ from the Eqs. above. We choose $\boldsymbol{R} = \boldsymbol{I}$.

- Note: if the MLE model is found using optimization methods or via the EM algorithm (see following slides), $\boldsymbol{R}$ is arbitrary and the columns of $\boldsymbol{W}$ not orthogonal.

  In this case, orthogonality can be enforced

  - as postprocessing or
  - by modifying the EM algorithm.


- Ahn, J. H. and J. H. Oh (2003). A constrained EM algorithm for principal component analysis. *Neural Computation* **15**(1), 57–65.


# Maximum Likelihood PCA

- If we consider no dimensionality reduction ($M = D$), then

$$\boldsymbol{U}_M = \boldsymbol{U}, \quad \boldsymbol{L}_M = \boldsymbol{L},$$

and using $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}$, $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$, we see that the covariance $\boldsymbol{C}$ of the marginal distribution for $\boldsymbol{x}$ becomes

$$\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I} = \boldsymbol{U}\left(\boldsymbol{L} - \sigma^2\boldsymbol{I}\right)^{1/2} \boldsymbol{R}\boldsymbol{R}^T \left(\boldsymbol{L} - \sigma^2\boldsymbol{I}\right)^{1/2} \boldsymbol{U}^T + \sigma^2\boldsymbol{I} = \boldsymbol{U}\boldsymbol{L}\boldsymbol{U}^T = \boldsymbol{S}$$

- This result is the standard MLE solution for an unconstrained Gaussian distribution in which the covariance matrix is given by the sample covariance.

# Efficient Evaluation of PPCA Density

- Since $\boldsymbol{C}$ is not full rank, we use the matrix inversion lemma:

$$\boldsymbol{C}^{-1} = \frac{1}{\sigma^2}\left[ I - \boldsymbol{W}\left(\boldsymbol{W}^T\boldsymbol{W} + \sigma^2 \boldsymbol{I}\right)^{-1}\boldsymbol{W}^T \right]$$

- Plugging the MLE estimates for $\boldsymbol{W}, \sigma^2$ gives: $\boldsymbol{W} = \boldsymbol{U}_M\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2}$

$$\boldsymbol{C}^{-1} = \frac{1}{\sigma^2}\left[ \boldsymbol{I} - \boldsymbol{U}_M\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2}\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I} + \sigma^2\boldsymbol{I}\right)^{-1}\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2}\boldsymbol{U}_M^T \right]$$

$$= \frac{1}{\sigma^2}\left[ \boldsymbol{I} - \boldsymbol{U}_M\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2}\boldsymbol{L}_M^{-1}\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2}\boldsymbol{U}_M^T \right]$$

$$= \frac{1}{\sigma^2}\left[ \boldsymbol{I} - \boldsymbol{U}_M \, diag\left(1 - \frac{\sigma^2}{\lambda_j}\right)\boldsymbol{U}_M^T \right]$$

- Similarly: $\log\left|\boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}\right| = (D-M)\log\sigma^2 + \sum_{i=1}^{M}\log\lambda_i$

Use the matrix inversion lemma $det(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T) = det(\boldsymbol{W}^{-1} + \boldsymbol{V}^T\boldsymbol{A}^{-1}\boldsymbol{U})det\boldsymbol{W}det\boldsymbol{A}$

# Efficient Evaluation of PPCA Density

- Since $\boldsymbol{C}$ is not full rank, we can use the matrix inversion lemma:

$$|\boldsymbol{C}| = \left|\boldsymbol{W}\boldsymbol{W}^T + \sigma^2 \boldsymbol{I}_{D\times D}\right| = \left|\boldsymbol{U}_M \left(\boldsymbol{L}_M - \sigma^2 \boldsymbol{I}_{M\times M}\right)\boldsymbol{U}^T + \sigma^2 \boldsymbol{I}_{D\times D}\right| \Rightarrow$$

$$|\boldsymbol{C}| = \left|\left(\boldsymbol{L}_M - \sigma^2 \boldsymbol{I}_{M\times M}\right)^{-1} + \boldsymbol{U}^T \sigma^{-2} \boldsymbol{I}_{M\times M}\boldsymbol{U}\right|\left|\left(\boldsymbol{L}_M - \sigma^2 \boldsymbol{I}_{M\times M}\right)\right|\left|\sigma^2 \boldsymbol{I}_{D\times D}\right|$$

- Here we used the MLE value for $\boldsymbol{W} = \boldsymbol{U}_M \left(\boldsymbol{L}_M - \sigma^2 \boldsymbol{I}\right)^{1/2}$ and the matrix inversion lemma $det(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T) = det(\boldsymbol{W}^{-1} + \boldsymbol{V}^T\boldsymbol{A}^{-1}\boldsymbol{U})det\boldsymbol{W}det\boldsymbol{A}$

- Taking logs and evaluating the above determinants gives:

$$\log|\boldsymbol{C}| = \left(\sum_{i=1}^{M}\log\lambda_i - M\log\sigma^2 - \sum_{i=1}^{M}\log\left(\lambda_i - \sigma^2\right)\right) + \sum_{i=1}^{M}\log\left(\lambda_i - \sigma^2\right) + D\log\sigma^2$$

$$\log\left|\boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}\right| = (D-M)\log\sigma^2 + \sum_{i=1}^{M}\log\lambda_i$$

# Maximum Likelihood PCA

- PCA is generally expressed as a projection of points from the $D$-dimensional dataspace onto an $M$-dimensional subspace.
- Probabilistic PCA is seen as the mapping $\boldsymbol{z} \rightarrow \boldsymbol{x}, \boldsymbol{x} = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon}$.
- The reverse mapping $\boldsymbol{x} \rightarrow \boldsymbol{z}$ is computed using the posterior:

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{M}^{-1}\boldsymbol{W}^{T}(\boldsymbol{x} - \boldsymbol{\mu}), \sigma^{2}\boldsymbol{M}^{-1}\right), \quad \boldsymbol{M} = \boldsymbol{W}^{T}\boldsymbol{W} + \sigma^{2}\boldsymbol{I}$$

- Every point in data space is characterized by its posterior mean and covariance in latent space:

$$\mathbb{E}\left[\boldsymbol{z} \mid \boldsymbol{x}\right] = \boldsymbol{M}^{-1}\boldsymbol{W}_{ML}^{T}(\boldsymbol{x} - \overline{\boldsymbol{x}}), \operatorname{var}\left[\boldsymbol{z} \mid \boldsymbol{x}\right] = \sigma^{2}\boldsymbol{M}^{-1}$$

- This projects to a point in data space given by:

$$\boldsymbol{W}\mathbb{E}\left[\boldsymbol{z} \mid \boldsymbol{x}\right] + \boldsymbol{\mu}$$

- This is identical form as regularized linear regression.

# Maximum Likelihood PCA vs Standard PCA

- In the limit $\sigma^2 \to 0$, the posterior mean becomes:

$$\mathbb{E}\left[\boldsymbol{z} \mid \boldsymbol{x}\right] = \boldsymbol{M}^{-1}\boldsymbol{W}_{ML}^{T}(\boldsymbol{x} - \overline{\boldsymbol{x}}),\ \boldsymbol{M} = \boldsymbol{W}_{ML}^{T}\boldsymbol{W}_{ML} + \sigma^2\boldsymbol{I} \to \boldsymbol{W}_{ML}^{T}\boldsymbol{W}_{ML} \Rightarrow$$

$$\mathbb{E}\left[\boldsymbol{z} \mid \boldsymbol{x}\right] = \left(\boldsymbol{W}_{ML}^{T}\boldsymbol{W}_{ML}\right)^{-1}\boldsymbol{W}_{ML}^{T}(\boldsymbol{x} - \overline{\boldsymbol{x}})$$

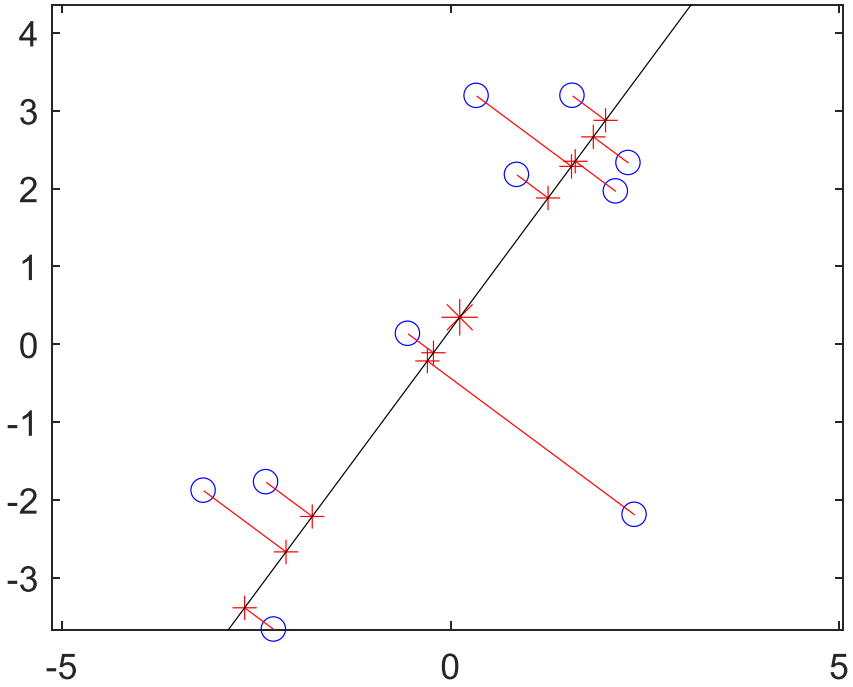
- Now substitute $\boldsymbol{W}_{ML} = \boldsymbol{U}_M\left(\boldsymbol{L}_M - \sigma^2\boldsymbol{I}\right)^{1/2}\boldsymbol{R} = \boldsymbol{U}_M\boldsymbol{L}_M^{1/2}$ for $\sigma^2 \to 0$ with $\boldsymbol{R} = \boldsymbol{I}$ (for consistency with PCA). Then

$$\mathbb{E}\left[\boldsymbol{z} \mid \boldsymbol{x}\right] = \left(\boldsymbol{W}_{ML}^{T}\boldsymbol{W}_{ML}\right)^{-1}\boldsymbol{W}_{ML}^{T}(\boldsymbol{x} - \overline{\boldsymbol{x}}) = \boldsymbol{L}_M^{-1/2}\boldsymbol{U}_M^{T}(\boldsymbol{x} - \overline{\boldsymbol{x}})$$

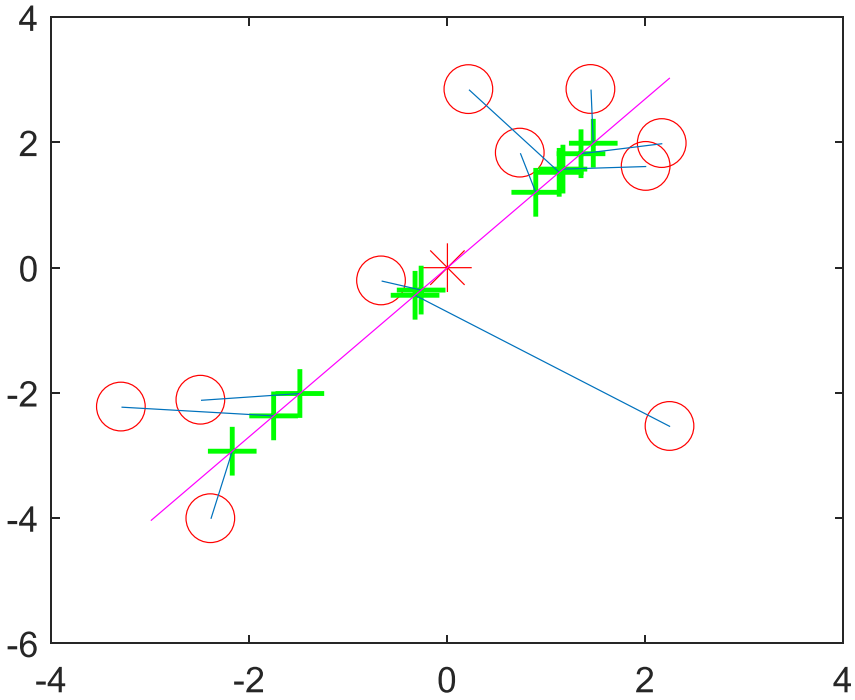
- This is an orthogonal projection of the data point onto the latent space, i.e. for the limit $\sigma^2 \to 0$, we recover the standard PCA model. The posterior covariance $\mathrm{var}\left[\boldsymbol{z} \mid \boldsymbol{x}\right] = \sigma^2\boldsymbol{M}^{-1} \to 0$ and the density becomes singular.
- For $\sigma^2 > 0$, the latent projection is shifted towards the origin, relative to the orthogonal projection.

# PPCA Versus PCA

- Consider PCA and PPCA where $D = 2$ and $M = 1$. The red star is the data mean. In PCA the points are orthogonally projected onto the line. In PPCA the projection is no longer orthogonal and the reconstructions are shrunk towards the data mean (red star).




Run pcaDemo2d
From PMTK3




Run ppcaDemo2d
From PMTK3

# Probabilistic PCA: Number of DOF

- PPCA defines a multivariate Gaussian distribution in which the number of DOF (independent parameters) can be controlled while still capturing the dominant correlations in the data.

- A general Gaussian distribution has $D(D+1)/2$ independent parameters in its covariance matrix and $D$ parameters in its mean.

  - The number of parameters scales as $D^2$.

- For a diagonal covariance matrix we have $D$ independent parameters and the number of parameters grows linearly. However, the variables are independent and hence this model cannot express any correlations between them.

# Probabilistic PCA: Number of DOF

- In PPCA, the $M$ most significant correlations are captured while the total number of parameters grows linearly with $D$.
- We can see this by evaluating the DOF in PPCA:
  - The covariance $\boldsymbol{C}$ depends on $\boldsymbol{W}$ ($D \times M$), and $\sigma^2$: $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^T + \sigma^2 \boldsymbol{I}$ total parameters $DM + 1$.
  - We need to subtract the redundancy associated with rotations of the coordinate system in the latent space.
  - $\boldsymbol{R}$ is $M \times M$. In the 1st column there are $M - 1$ independent parameters (must be normalized). In the 2nd column there are $M - 2$ independent parameters (normalized & orthogonal to the 1st column), etc. $\boldsymbol{R}$ has a total of $M(M-1)/2$ independent parameters.
- The number of degrees of freedom in $\boldsymbol{C}$ grows linearly with $D$

$$D \times M + 1 - M \times (M-1)/2$$

# Probabilistic PCA: Number of DOF

- The number of DOF in $\boldsymbol{C}$ grows linearly in $D$ for a given $M$:

$$D \times M + 1 - M \times (M - 1)/2$$

- $M = D - 1$: recover the standard result for a full covariance Gaussian.
  - The variance along $D - 1$ linearly independent directions is controlled by the columns of $\boldsymbol{W}$, and
  - The variance along the remaining direction is given by $\sigma^2$.
- $M = 0$: equivalent to the isotropic covariance case.

# EM Algorithm for PCA

# EM Algorithm for PCA

- The PPCA model involves a marginalization over a continuous latent space $\boldsymbol{z}$. For each $\boldsymbol{x}_n$ there is a corresponding $\boldsymbol{z}_n$.
- Use EM to find the MLE of the model parameters.
- For high $D$, there are advantages using iteratively EM vs. working directly with the sample covariance.
- This approach can be extended to factor analysis for which there is no closed-form solution.
- Can also be used when values are missing & for mixture models.
- EM requires the complete-data log likelihood function:

$$\ln p\left(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\mu},\boldsymbol{W},\sigma^2\right)=\sum_{n=1}^{N}\left\{\ln p\left(\boldsymbol{x}_n\mid\boldsymbol{z}_n\right)+\ln p\left(\boldsymbol{z}_n\right)\right\}$$

- In the following, we substitute $\mu$ with the sample mean $\overline{\boldsymbol{x}}$.

# EM Algorithm for PCA

- We first take the expectation of the complete-data log likelihood with respect to the posterior distribution of the latent distribution evaluated using 'old' parameter values.

- Maximization of this expected complete data log likelihood then yields the 'new' parameter values.

$$\ln p\left(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2\right) = \sum_{n=1}^{N} \left\{\ln p\left(\boldsymbol{x}_n \mid \boldsymbol{z}_n\right) + \ln p\left(\boldsymbol{z}_n\right)\right\}$$

- The $n^{\text{th}}$ row of $\boldsymbol{Z}$ is given by $\boldsymbol{z}_n$ .

- Recall that $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$ , $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$

- We can now write the expectation with respect to the posterior distribution over the latent variables.

# EM Algorithm for PCA

$$\mathbb{E}\left[\ln p\left(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\mu},\boldsymbol{W},\sigma^2\right)\right]=-\sum_{n=1}^{N}\left\{\begin{aligned}&\frac{D}{2}\ln\left(2\pi\sigma^2\right)+\frac{1}{2}Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\right)\\&+\frac{1}{2\sigma^2}\left\|\boldsymbol{x}_n-\boldsymbol{\mu}\right\|^2-\frac{1}{\sigma^2}\mathbb{E}\left[\boldsymbol{z}_n\right]^T\boldsymbol{W}^T\left(\boldsymbol{x}_n-\boldsymbol{\mu}\right)\\&+\frac{1}{2\sigma^2}Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\boldsymbol{W}^T\boldsymbol{W}\right)+\frac{M}{2\ln\left(2\pi\right)}\end{aligned}\right\}$$

- E-Step: We use the old parameters to evaluate:

$$\mathbb{E}\left[\boldsymbol{z}_n\right]=\boldsymbol{M}^{-1}\boldsymbol{W}^T\left(\boldsymbol{x}_n-\bar{\boldsymbol{x}}\right)\qquad\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]=\sigma^2\boldsymbol{M}^{-1}+\mathbb{E}\left[\boldsymbol{z}_n\right]\mathbb{E}\left[\boldsymbol{z}_n\right]^T$$

- This follows directly from

$$p(\boldsymbol{z}\mid\boldsymbol{x})=\mathcal{N}\left(\boldsymbol{z}\mid\boldsymbol{M}^{-1}\boldsymbol{W}^T(\boldsymbol{x}-\boldsymbol{\mu}),\sigma^2\boldsymbol{M}^{-1}\right),\quad\boldsymbol{M}=\boldsymbol{W}^T\boldsymbol{W}+\sigma^2\boldsymbol{I}$$

together with the standard result

$$\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]=\mathrm{cov}\left[\boldsymbol{z}_n\right]+\mathbb{E}\left[\boldsymbol{z}_n\right]\mathbb{E}\left[\boldsymbol{z}_n\right]^T$$

# EM Algorithm for PCA

$$\mathbb{E}\left[\ln p\left(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\mu},\boldsymbol{W},\sigma^2\right)\right]=-\sum_{n=1}^{N}\left\{\begin{aligned}&\frac{D}{2}\ln\left(2\pi\sigma^2\right)+\frac{1}{2}Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\right)\\&+\frac{1}{2\sigma^2}\left\|\boldsymbol{x}_n-\boldsymbol{\mu}\right\|^2-\frac{1}{\sigma^2}\mathbb{E}\left[\boldsymbol{z}_n\right]^T\boldsymbol{W}^T\left(\boldsymbol{x}_n-\boldsymbol{\mu}\right)\\&+\frac{1}{2\sigma^2}Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\boldsymbol{W}^T\boldsymbol{W}\right)+\frac{M}{2\ln\left(2\pi\right)}\end{aligned}\right\}$$

- M-Step: We maximize with respect to $\boldsymbol{W}$ and $\sigma^2$ keeping the posterior statistics fixed. We obtain:

$$\boldsymbol{W}_{new}=\left[\sum_{n=1}^{N}\left(\boldsymbol{x}_n-\bar{\boldsymbol{x}}\right)\mathbb{E}\left[\boldsymbol{z}_n\right]^T\right]\left[\sum_{n=1}^{N}\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\right]^{-1}$$

$$\sigma^2_{new}=\frac{1}{ND}\sum_{n=1}^{N}\left\{\begin{aligned}&\left\|\boldsymbol{x}_n-\bar{\boldsymbol{x}}\right\|^2-2\mathbb{E}\left[\boldsymbol{z}_n\right]^T\boldsymbol{W}_{new}^T\left(\boldsymbol{x}_n-\bar{\boldsymbol{x}}\right)\\&+Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\boldsymbol{W}_{new}^T\boldsymbol{W}_{new}\right)\end{aligned}\right\}$$

# Proof of the M-Step Equations

$$\mathbb{E}\left[\ln p\left(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\mu},\boldsymbol{W},\sigma^2\right)\right]=-\sum_{n=1}^{N}\left\{\begin{array}{l}\frac{D}{2}\ln\left(2\pi\sigma^2\right)+\frac{1}{2}Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\right)\\+\frac{1}{2\sigma^2}\left\|\boldsymbol{x}_n-\boldsymbol{\mu}\right\|^2-\frac{1}{\sigma^2}\mathbb{E}\left[\boldsymbol{z}_n\right]^T\boldsymbol{W}^T\left(\boldsymbol{x}_n-\boldsymbol{\mu}\right)\\+\frac{1}{2\sigma^2}Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\boldsymbol{W}^T\boldsymbol{W}\right)+\frac{M}{2\ln\left(2\pi\right)}\end{array}\right\}$$

- The two M-equations are derived by setting the derivatives wrt $\boldsymbol{W}$ and $\sigma^2$ equal to zero:

$$\frac{\partial\mathbb{E}\left[\ln p\left(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\mu},\boldsymbol{W},\sigma^2\right)\right]}{\partial\boldsymbol{W}}=\sum_{n=1}^{N}\left\{\frac{1}{\sigma^2}\left(\boldsymbol{x}_n-\boldsymbol{\mu}\right)\mathbb{E}\left[\boldsymbol{z}_n\right]^T-\frac{1}{\sigma^2}\boldsymbol{W}\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\right\}=0$$

$$\frac{\partial\mathbb{E}\left[\ln p\left(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\mu},\boldsymbol{W},\sigma^2\right)\right]}{\partial\sigma^2}=\sum_{n=1}^{N}\left\{\begin{array}{l}-\frac{D}{2\sigma^2}-\frac{1}{\sigma^4}\mathbb{E}\left[\boldsymbol{z}_n\right]^T\boldsymbol{W}^T\left(\boldsymbol{x}_n-\boldsymbol{\mu}\right)+\frac{1}{2\sigma^4}\left\|\boldsymbol{x}_n-\boldsymbol{\mu}\right\|^2\\+\frac{1}{2\sigma^4}Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\boldsymbol{W}^T\boldsymbol{W}\right)\end{array}\right\}=0$$

- Here, we used $\frac{\partial}{\partial\boldsymbol{A}}Tr(\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T)=\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{B}^T),\ \frac{\partial}{\partial\boldsymbol{A}}Tr(\boldsymbol{A}\boldsymbol{B})=\boldsymbol{B}^T$

# EM Algorithm for PCA

- Initialize the parameters

- Compute the sufficient statistics of the latent space posterior distribution in the E-Step

$$\mathbb{E}\left[\boldsymbol{z}_n\right] = \boldsymbol{M}^{-1}\boldsymbol{W}^T\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right) \qquad \mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right] = \sigma^2\boldsymbol{M}^{-1} + \mathbb{E}\left[\boldsymbol{z}_n\right]\mathbb{E}\left[\boldsymbol{z}_n^T\right]$$

- Revise the parameter values in the M-Step.

$$\boldsymbol{W}_{new} = \left[\sum_{n=1}^{N}\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)\mathbb{E}\left[\boldsymbol{z}_n\right]^T\right]\left[\sum_{n=1}^{N}\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\right]^{-1}$$

$$\sigma_{new}^2 = \frac{1}{ND}\sum_{n=1}^{N}\left\{\begin{array}{l}\left\|\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right\|^2 - 2\left[\boldsymbol{z}_n\right]^T\boldsymbol{W}_{new}^T\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right) \\ +Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\boldsymbol{W}_{new}^T\boldsymbol{W}_{new}\right)\end{array}\right\}$$

# EM Algorithm for PCA: Computational Cost

- Each cycle of the EM algorithm can be computationally more efficient than conventional PCA in high dimensions.
- The eigen-decomposition of the covariance matrix requires $\mathcal{O}(D^3)$ computation. If interested only in the first $M$ eigenvectors, we can use algorithms that are $\mathcal{O}(MD^2)$.
- However, the evaluation of the covariance matrix itself

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}_n - \overline{\boldsymbol{x}}\right)\left(\boldsymbol{x}_n - \overline{\boldsymbol{x}}\right)^T$$

  takes $\mathcal{O}(ND^2)$ computations.
- Algorithms such as the snapshot method (Sirovich,1987), assume that the eigenvectors are linear combinations of the data vectors and avoid direct evaluation of the covariance matrix but are $\mathcal{O}(N^3)$ and hence unsuited to large data sets.


- Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. *Quarterly Applied Mathematics* **45**(3), 561–590.


# EM Algorithm for PCA: Computational Cost

- The EM algorithm described here does not construct the covariance matrix explicitly.

- Instead, the most computationally demanding steps are those involving sums over the data set that are $\mathcal{O}(NDM)$.

$$\boldsymbol{W}_{new} = \left[\sum_{n=1}^{N}\left(\boldsymbol{x}_n - \overline{\boldsymbol{x}}\right)\mathbb{E}\left[\boldsymbol{z}_n\right]^T\right]\left[\sum_{n=1}^{N}\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\right]^{-1}$$

- For large $D$, and $M << D, \mathcal{O}(NDM)$ is much better compared to $\mathcal{O}(ND^2)$ (that is needed to compute the sample covariance)

  This offsets the iterative nature of the EM algorithm.

# EM Algorithm for PCA: Online form

- The EM algorithm can be implemented in an on-line form: each $D$-dim data point is read/processed/then discarded before the next data point is considered.

- To see this, note that the quantities evaluated in the E-Step (an $M$-dimensional vector and an $M \times M$ matrix) can be computed for each data point separately.

$$\mathbb{E}\left[\boldsymbol{z}_n\right] = \boldsymbol{M}^{-1}\boldsymbol{W}^T\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right) \qquad \mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right] = \sigma^2\boldsymbol{M}^{-1} + \mathbb{E}\left[\boldsymbol{z}_n\right]\mathbb{E}\left[\boldsymbol{z}_n\right]^T$$

- In the $M$ –Step, we accumulate the sums over data points incrementally - advantageous if both $N$ & $D$ are large.

$$\boldsymbol{W}_{new} = \left[\sum_{n=1}^{N}\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)\mathbb{E}\left[\boldsymbol{z}_n\right]^T\right]\left[\sum_{n=1}^{N}\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\right]^{-1}$$

$$\sigma^2_{new} = \frac{1}{ND}\sum_{n=1}^{N}\left\{\begin{array}{l}\left\|\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right\|^2 - 2\mathbb{E}\left[\boldsymbol{z}_n\right]^T\boldsymbol{W}_{new}^T\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right) \\ +Tr\left(\mathbb{E}\left[\boldsymbol{z}_n\boldsymbol{z}_n^T\right]\boldsymbol{W}_{new}^T\boldsymbol{W}_{new}\right)\end{array}\right\}$$

# EM Algorithm for PCA: Missing Values

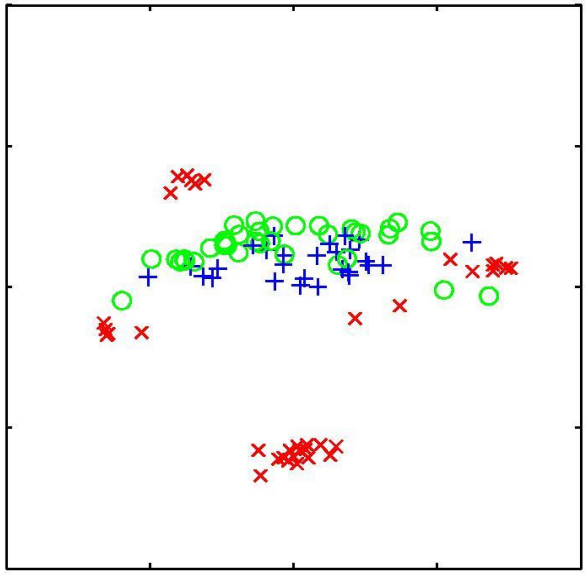
- Because we now have a fully probabilistic model for PCA, we can deal with missing data, provided that it is missing at random, by marginalizing over the distribution of the unobserved variables.

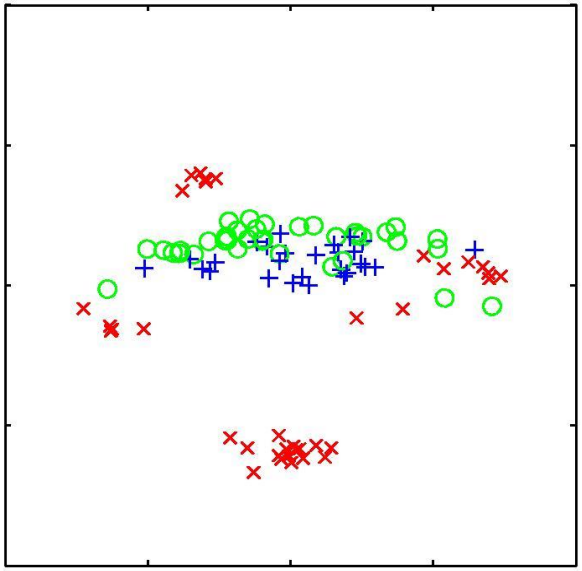
- Again these missing values can be treated using the EM algorithm.

- We give an example of the use of this approach for data visualization in an example next.

# EM Algorithm for PCA: Missing Values

- PPCA visualization for the first 100 data points of the oil flow data set.
- Left: the posterior mean projections of the data points on the principal subspace.
- Right: randomly omitting 30% of the variable values and using EM to handle the missing values. Even though each data point has at least one missing measurement, the plot is similar to the one obtained without missing values.





Matlab Implementation

# EM Algorithm for PCA: Limit $\sigma^2 \to 0$

- When $\sigma^2 \to 0$, EM corresponds to standard PCA (Roweis, 1998)
- Defining $\tilde{\boldsymbol{X}}$ a matrix of size $N \times D$ whose $n^{\text{th}}$ row is given by $\boldsymbol{x}_n - \bar{\boldsymbol{x}}$.
- Defining a matrix $\Omega$ of size $M \times N$ whose $n^{\text{th}}$ column is given by the vector $\mathbb{E}[\boldsymbol{z}_n]$.
- The E-Step for $\sigma^2 \to 0$ becomes

$$\mathbb{E}\left[\boldsymbol{z}_n\right] = \boldsymbol{M}^{-1}\boldsymbol{W}^T\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right) \Rightarrow \quad \boldsymbol{\Omega} = \left(\boldsymbol{W}_{old}^T\boldsymbol{W}_{old}\right)^{-1}\boldsymbol{W}_{old}^T\tilde{\boldsymbol{X}}^T$$

This is simply the orthogonal projection of the data points on the current estimate for the principal subspace.


- Roweis, S. (1998). EM algorithms for PCA and SPCA. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 626–632. MIT Press.

# EM Algorithm for PCA: Limit $\sigma^2 \to 0$

- When $\sigma^2 \to 0$, EM corresponds to standard PCA
- $\tilde{\boldsymbol{X}}$ an $N \times D$ matrix whose $n^{\text{th}}$ row is given by $\boldsymbol{x}_n - \bar{\boldsymbol{x}}$.
- $\Omega$ a matrix $M \times N$ whose $n^{\text{th}}$ column is given by $\mathbb{E}[\boldsymbol{z}_n]$.
- Noting that

$$\sigma^2 \to 0 \;\Rightarrow\; \mathbb{E}\left[\boldsymbol{z}_n \boldsymbol{z}_n^T\right] = \sigma^2 \boldsymbol{M}^{-1} + \mathbb{E}\left[\boldsymbol{z}_n\right]\mathbb{E}\left[\boldsymbol{z}_n\right]^T \to \mathbb{E}\left[\boldsymbol{z}_n\right]\mathbb{E}\left[\boldsymbol{z}_n\right]^T$$

the $M$ −Step takes the form:

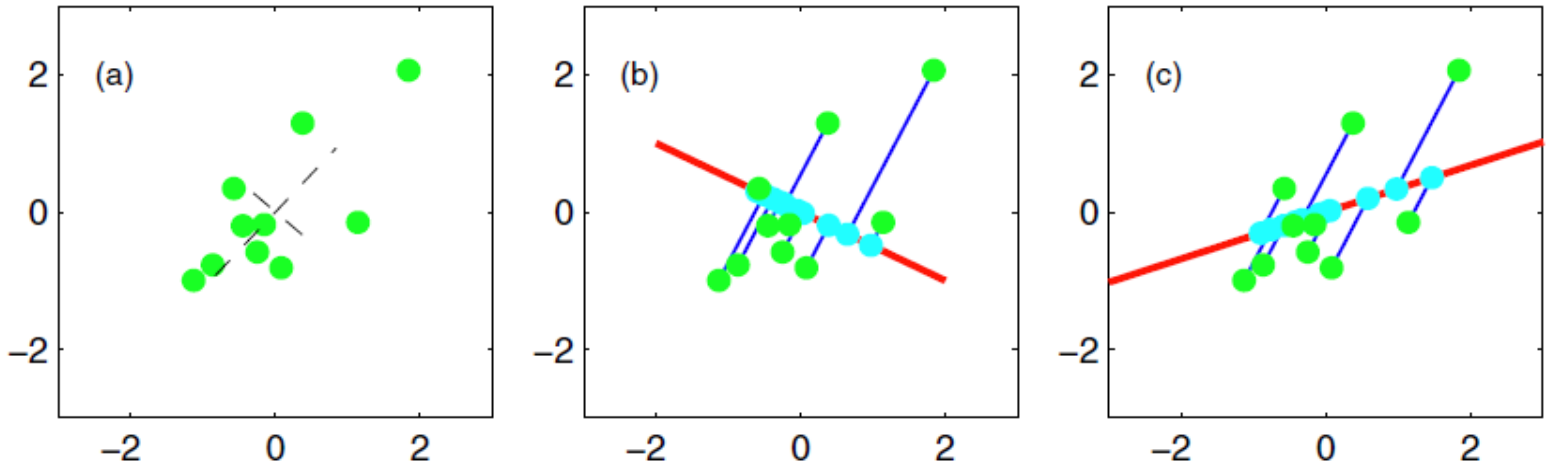
$$\boldsymbol{W}_{new} = \left[\sum_{n=1}^{N}\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)\mathbb{E}\left[\boldsymbol{z}_n\right]^T\right]\left[\sum_{n=1}^{N}\mathbb{E}\left[\boldsymbol{z}_n \boldsymbol{z}_n^T\right]\right]^{-1} \Rightarrow \quad \boldsymbol{W}_{new} = \tilde{\boldsymbol{X}}^T \boldsymbol{\Omega}^T (\boldsymbol{\Omega}\boldsymbol{\Omega}^T)^{-1}$$

Re-estimation of the principal subspace minimizing the squared reconstruction errors in which the projections are fixed (see interpretation next and also here)

# EM Algorithm for PCA: Example, $D = 2, M = 1$

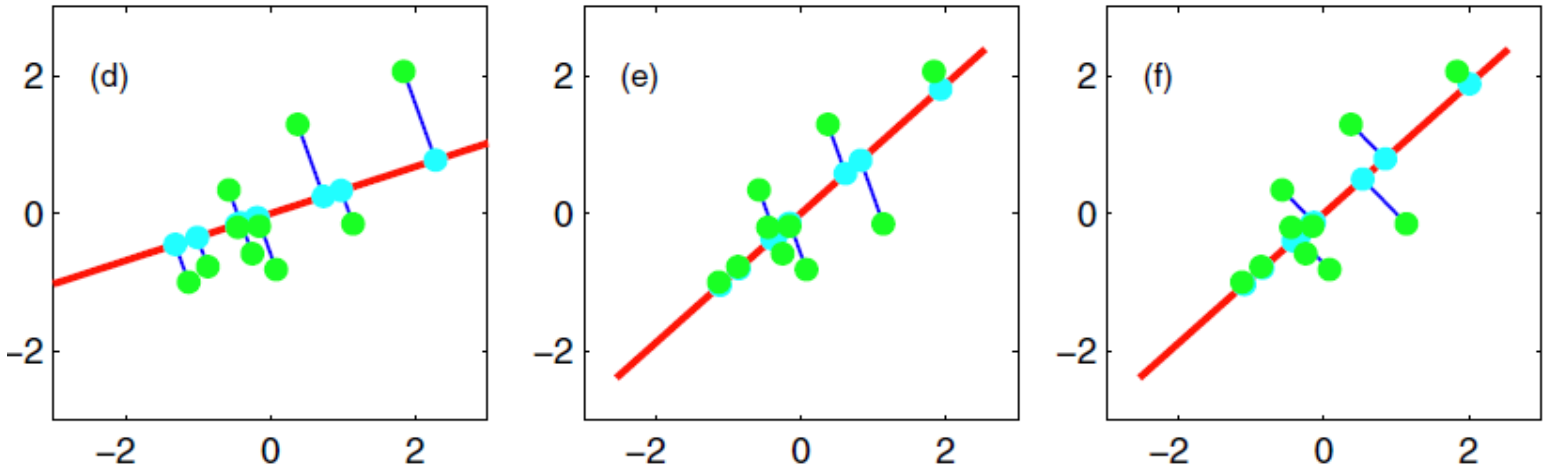
- Synthetic data illustrating the EM algorithm for PCA
  - (a) A data set $\boldsymbol{X}$ with the data points (green), together with the true principal components (eigenvectors scaled by the square roots of the eigenvalues).
  - (b) Initial configuration of the principal subspace defined by $\boldsymbol{W}$ (red) together with the projections of the latent points $\boldsymbol{Z}$ into the data space, given by $\boldsymbol{ZW}^T$ (cyan)
  - (c) After one M-Step, the latent space has been updated with $\boldsymbol{Z}$ held fixed.

Matlab Implementation



# EM Algorithm for PCA: Example, $D = 2, M = 1$

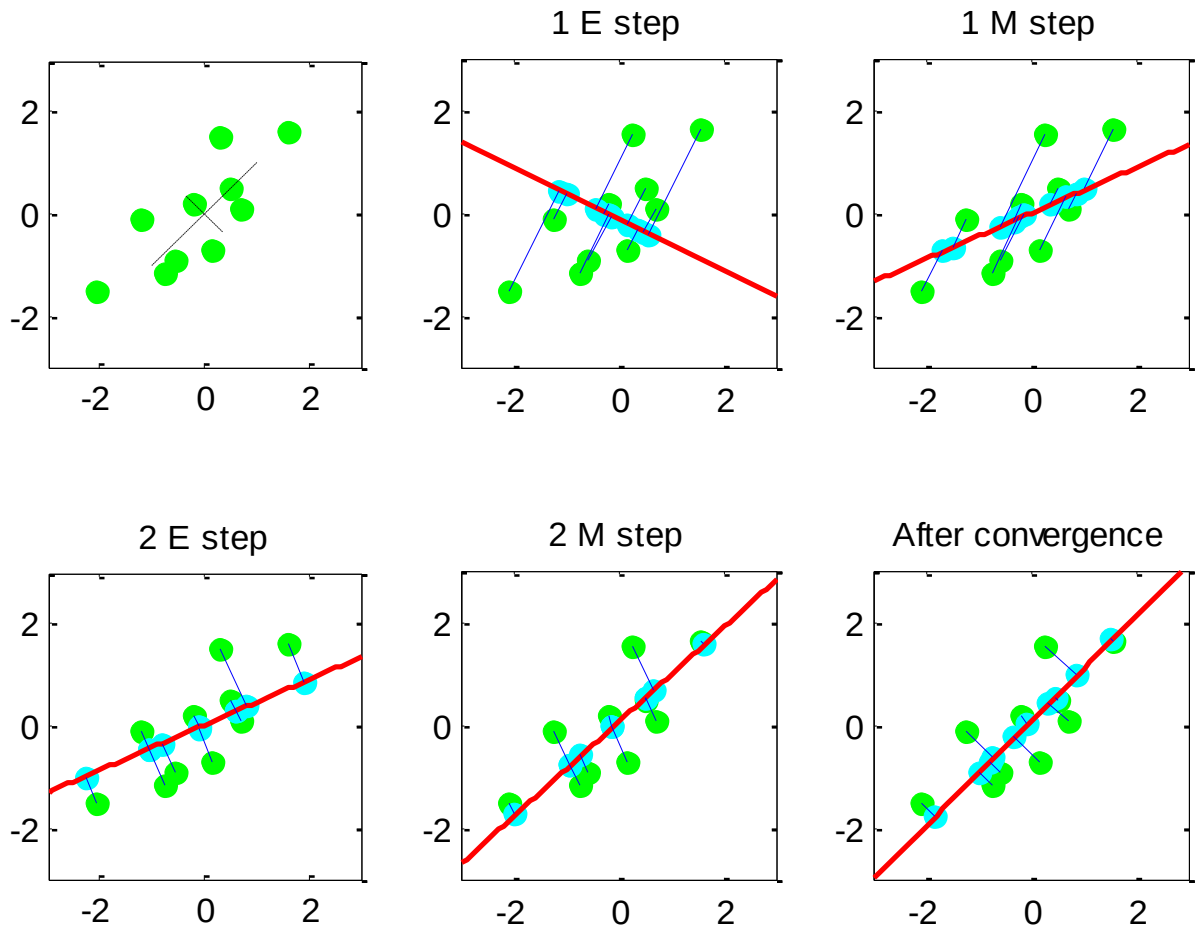
- Synthetic data illustrating the EM algorithm for PCA
  - (d) After the successive E-Step, the values of $\boldsymbol{Z}$ have been updated, giving orthogonal projections, with $\boldsymbol{W}$ held fixed.
  - (e) After the second M-Step.
  - (f) The converged solution.




$$\boldsymbol{\Omega} = \left(\boldsymbol{W}_{old}^T \boldsymbol{W}_{old}\right)^{-1} \boldsymbol{W}_{old}^T \widetilde{\boldsymbol{X}}^T$$

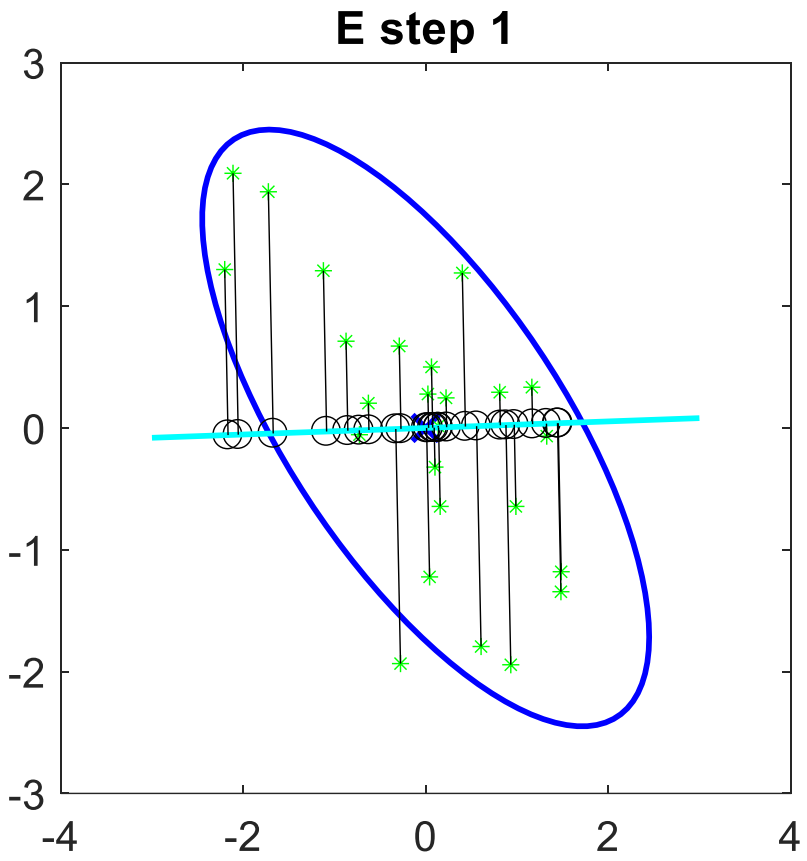
$$\boldsymbol{W}_{new} = \widetilde{\boldsymbol{X}}^T \boldsymbol{\Omega}^T \left(\boldsymbol{\Omega}\boldsymbol{\Omega}^T\right)^{-1}$$

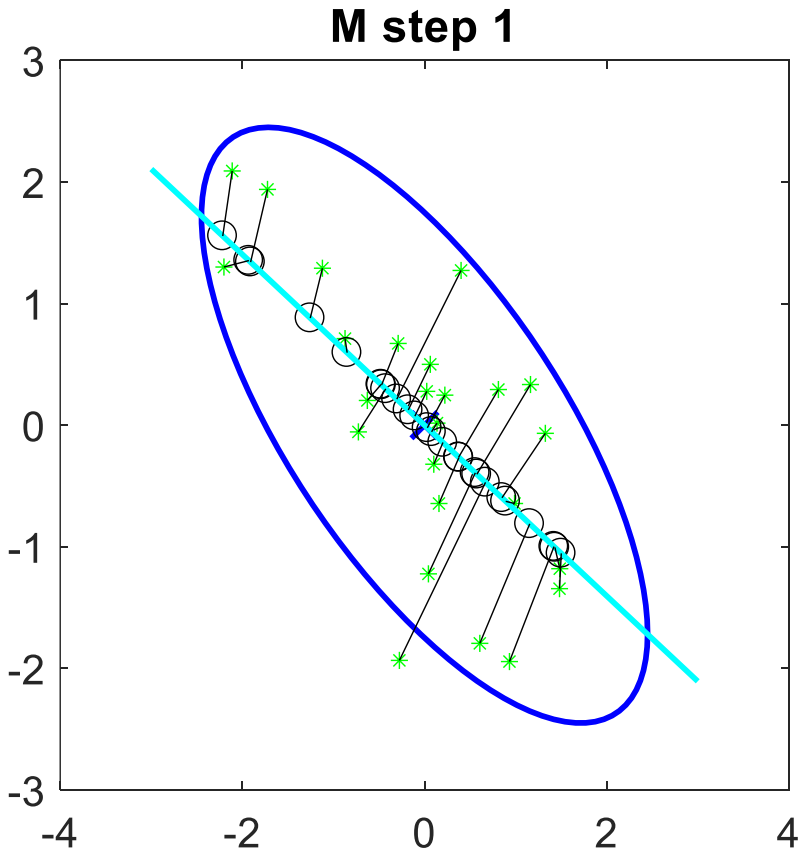
# EM Algorithm for PCA: Example, $D = 2, M = 1$




Matlab Implementation

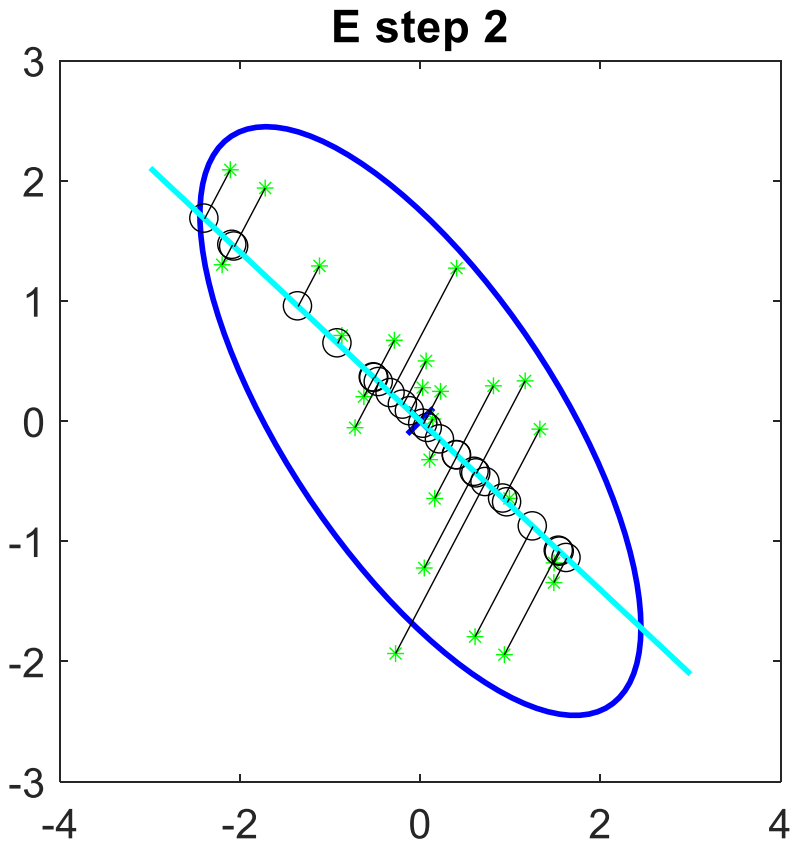
# EM Algorithm for PCA: Example, $D = 2, M = 1$

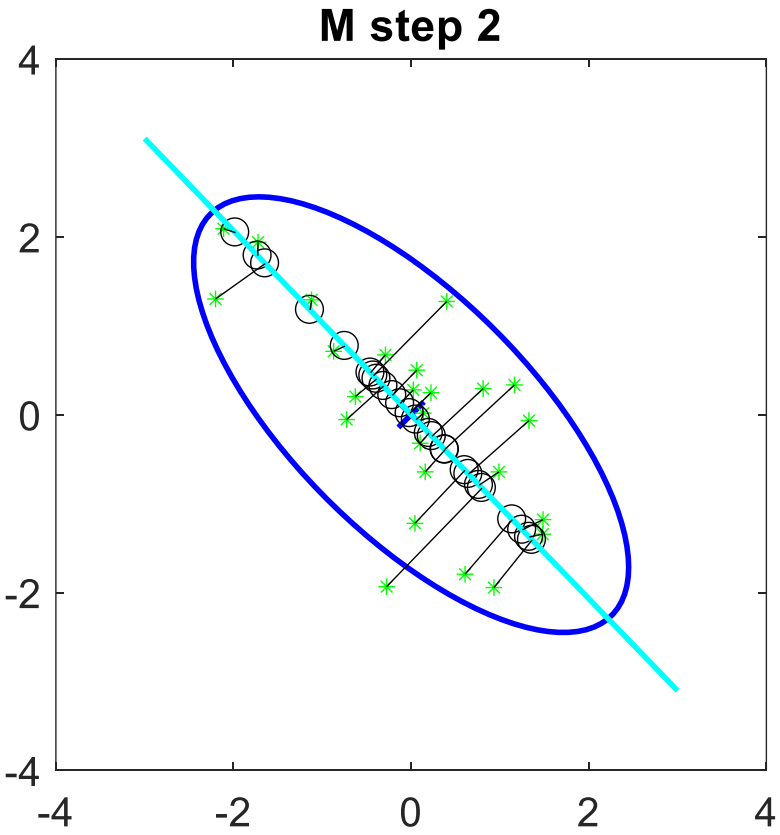



Run pcaEmStepByStep
From PMTK3

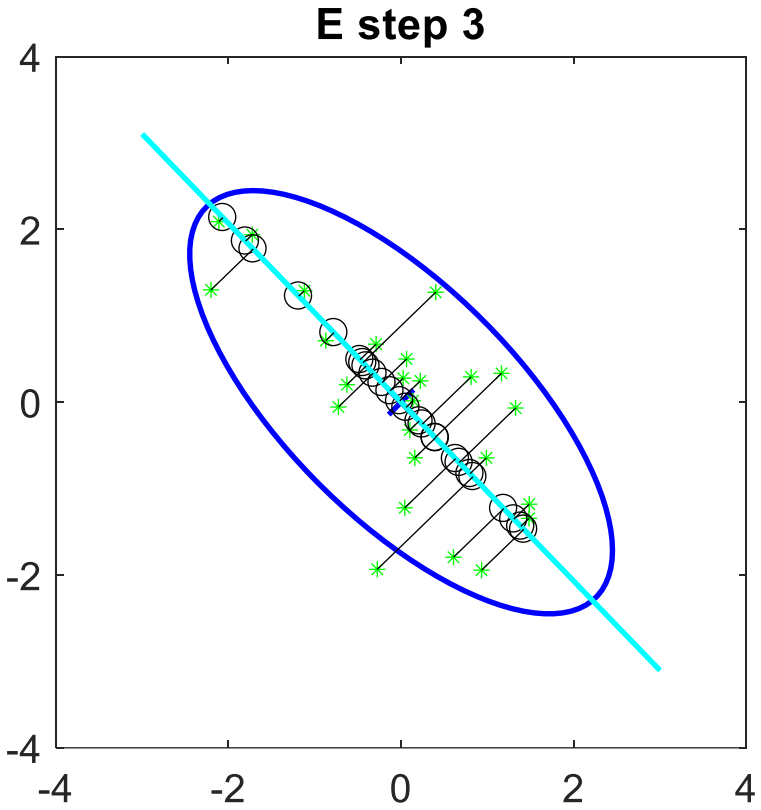
# EM Algorithm for PCA: Example, $D = 2, M = 1$

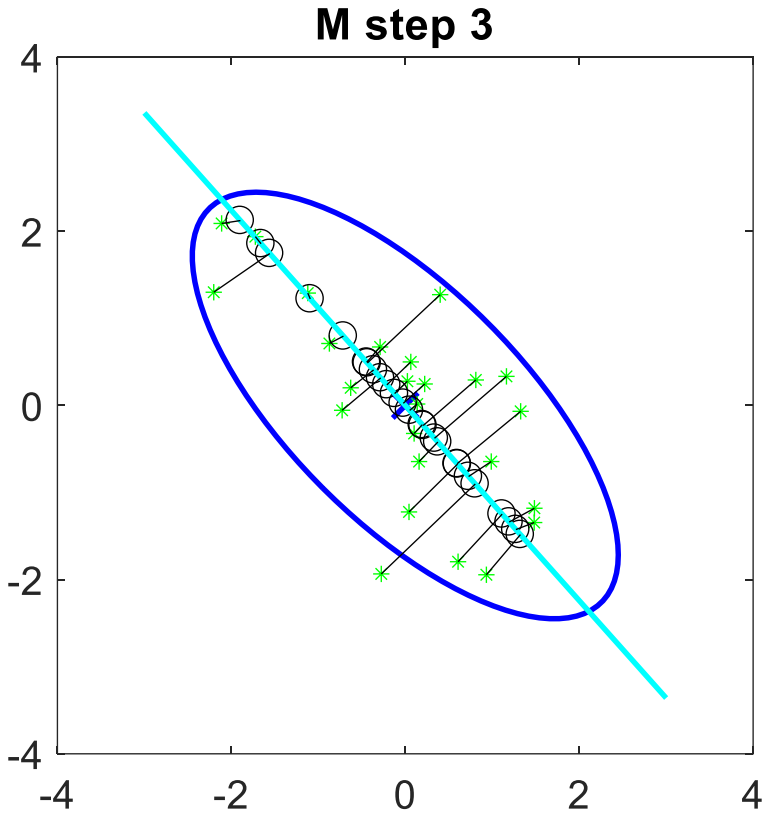



Run pcaEmStepByStep
From PMTK3

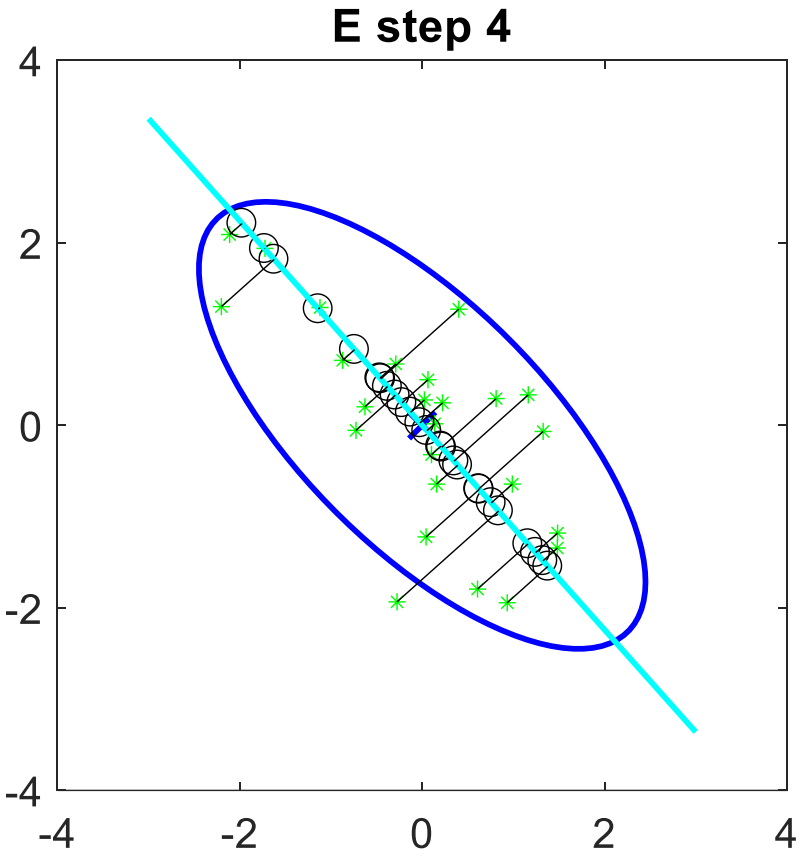
# EM Algorithm for PCA: Example, $D = 2, M = 1$

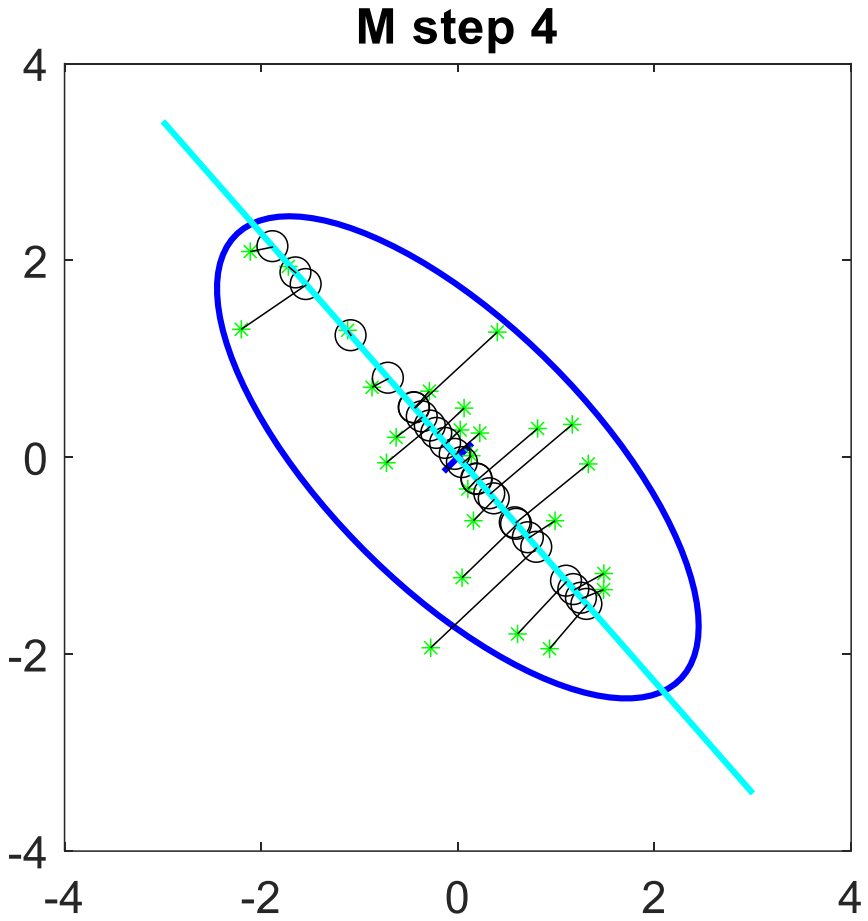



Run pcaEmStepByStep
From PMTK3

# EM Algorithm for PCA: Example, $D = 2, M = 1$




Run pcaEmStepByStep
From PMTK3

# EM Algorithm and Standard PCA Revisited

- Let $\boldsymbol{W}$ be $D \times M$ whose columns define a linear subspace of dimensionality $M$ embedded within a data set of dimensionality $D$.
- Let $\boldsymbol{\mu}$ a $D$-dimensional vector. We approximate the data points $\boldsymbol{x}_n, n = 1, .., N$ using a linear mapping from a set of $D$-dimensional vectors $\boldsymbol{z}_n$ as $\boldsymbol{W}\boldsymbol{z}_n + \boldsymbol{\mu}$.
- The sum of squares reconstruction error is:

$$J = \sum_{n=1}^{N} \left\| \boldsymbol{x}_n - \boldsymbol{\mu} - \boldsymbol{W}\boldsymbol{z}_n \right\|^2$$

- Minimizing wrt $\boldsymbol{\mu}$ gives:

$$0 = -\sum_{n=1}^{N} \left( \boldsymbol{x}_n - \boldsymbol{\mu} - \boldsymbol{W}\boldsymbol{z}_n \right) \Rightarrow \boldsymbol{\mu} = \bar{\boldsymbol{x}} - \boldsymbol{W}\bar{\boldsymbol{z}}$$

- This modifies the error as:

$$J = \sum_{n=1}^{N} \left\| \left( \boldsymbol{x}_n - \bar{\boldsymbol{x}} \right) - \boldsymbol{W} \left( \boldsymbol{z}_n - \bar{\boldsymbol{z}} \right) \right\|^2$$

# EM Algorithm and Standard PCA Revisited

- Let $\boldsymbol{X}$ be a $N \times D$ matrix with $n^{\text{th}}$ row $\left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right)^T$.
- Similarly let $\boldsymbol{Z}$ be a $N \times M$ matrix with $n^{\text{th}}$ row $\left(\boldsymbol{z}_n - \bar{\boldsymbol{z}}\right)^T$.
- The cost function can be written as:

$$J = \sum_{n=1}^{N} \left\| \left(\boldsymbol{x}_n - \bar{\boldsymbol{x}}\right) - \boldsymbol{W}\left(\boldsymbol{z}_n - \bar{\boldsymbol{z}}\right) \right\|^2 \Rightarrow \quad J = Tr\left\{ \left(\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{W}^T\right)\left(\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{W}^T\right)^T \right\}$$

- Setting the derivative wrt $\boldsymbol{Z}$ equal to $\boldsymbol{0}$ gives (using $\frac{\partial}{\partial \boldsymbol{A}} Tr(\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T) = \boldsymbol{A}(\boldsymbol{B} + \boldsymbol{B}^T)$ and $\frac{\partial}{\partial \boldsymbol{A}} Tr(\boldsymbol{A}\boldsymbol{B}) = \boldsymbol{B}^T, \frac{\partial}{\partial \boldsymbol{A}} Tr(\boldsymbol{A}^T\boldsymbol{B}) = \boldsymbol{B}$):

$$-2\boldsymbol{X}\boldsymbol{W} + 2\boldsymbol{Z}\boldsymbol{W}^T\boldsymbol{W} = 0 \Rightarrow \boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}\left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1}$$

This is simllar to the PCA E-Step: $\boldsymbol{\Omega} = \left(\boldsymbol{W}_{old}^T\boldsymbol{W}_{old}\right)^{-1}\boldsymbol{W}_{old}^T\tilde{\boldsymbol{X}}^T$

- Similarly derivative wrt $\boldsymbol{W}$ equal to $\boldsymbol{0}$ gives (analogous to the PCA M-Step: $\boldsymbol{W}_{new} = \tilde{\boldsymbol{X}}^T\boldsymbol{\Omega}^T(\boldsymbol{\Omega}\boldsymbol{\Omega}^T)^{-1}$)

$$-2\boldsymbol{X}^T\boldsymbol{Z} + 2\boldsymbol{W}\boldsymbol{Z}^T\boldsymbol{Z} = 0 \Rightarrow \boldsymbol{W} = \boldsymbol{X}^T\boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{Z}\right)^{-1}$$

# Nonlinear Latent Variable Models

# Independent Component Analysis

# Independent Component Analysis

- Setup:
  - Two people talking at the same time
  - Their voices recorded using two microphones
- Objective: to reconstruct the 2 signals separately "blindly" (we are given only the mixed data). We haven't observed
  - the original sources or the mixing coefficients
- Under some assumptions (no time delay and echoes)
  - the signals received by the microphone are linear combinations of the voice amplitudes
  - the coefficient of this linear combination are constant


- Cardoso, J.-F. (1998). Blind signal separation: statistical principles. *Proceedings of the IEEE* **9**(10), 2009–2025.
- Stone, J. V. (2004). *Independent Component Analysis: A Tutorial Introduction*. MIT Press.
- A Tutorial on Independent Component Analysis Jonathon Shlens
- A. Hyvarinen, E. Oja, Independent component analysis: algorithms and applications, Neural Networks 13 (2000) 411–430
- A. Ng, ICA, Lecture notes
- B. Poczos, Introduction to ICA, 2009


# Independent Component Analysis

- Model

$$x_1(t) = W_{11}z_1(t) + W_{12}z_2(t)$$
$$x_2(t) = W_{21}z_1(t) + W_{22}z_2(t)$$

- Observations

$$\begin{pmatrix} x_1(1) \\ x_2(1) \end{pmatrix}, \begin{pmatrix} x_1(2) \\ x_2(2) \end{pmatrix}, \ldots, \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

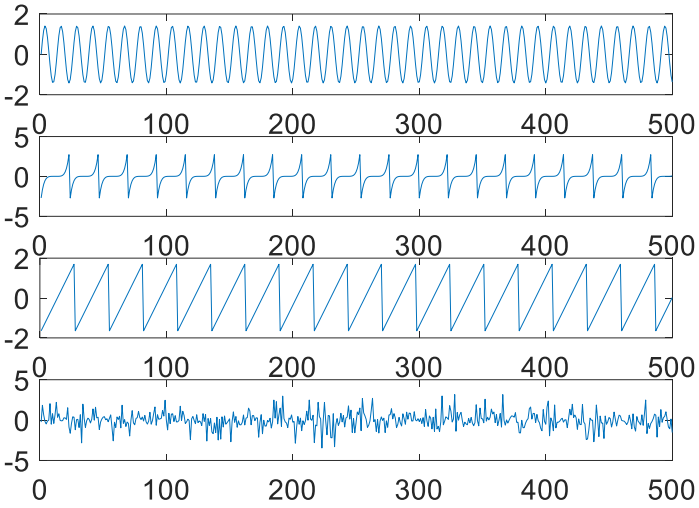
- Objective: Estimate

$$\big(z_1(1), z_2(1)\big), \ldots \big(z_1(t), z_2(t)\big)$$

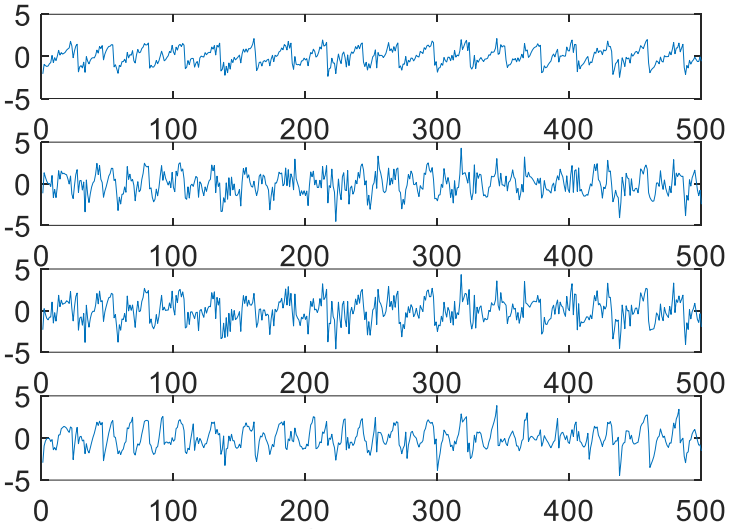
and the coefficients $W_{ij}$

# Independent Component Analysis

truth

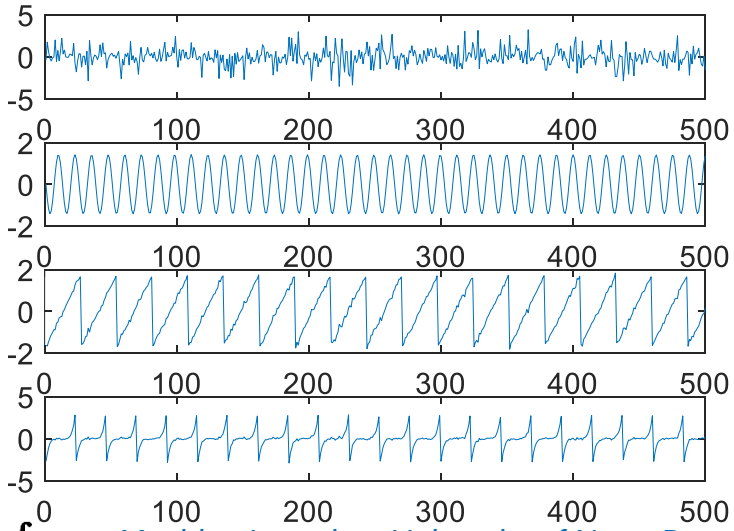



observed signals

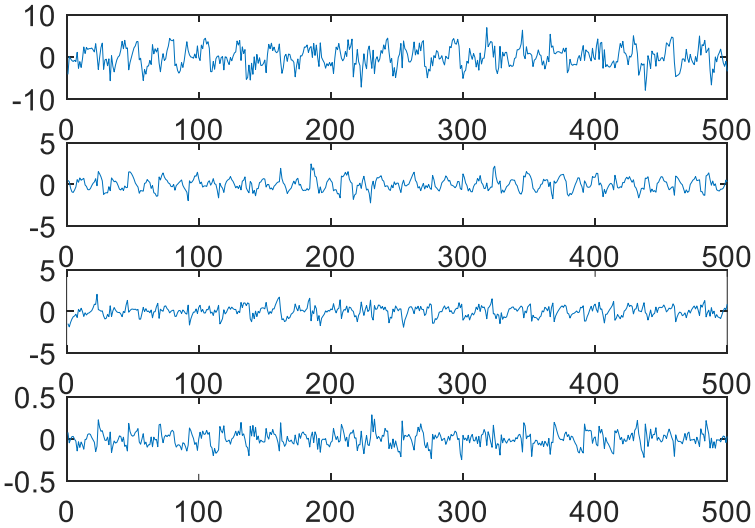



[Run icaDemo](#)
From [PMTK3](#)

ICA estimate




PCA estimate



# Independent Component Analysis

- Let $\boldsymbol{x_t} \in \mathbb{R}^D$ be the observed signal at the sensors at time $t$, and $\boldsymbol{z_t} \in \mathbb{R}^{\boldsymbol{L}}$ be the vector of source signals.

$$\boldsymbol{x_t} = \boldsymbol{W}\boldsymbol{z}_t + \boldsymbol{\varepsilon}_t$$

- Here $\boldsymbol{W} \in \mathbb{R}^{D \times \boldsymbol{L}}$, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$. Often $\boldsymbol{\Psi} = \mathbf{0}$.
- We treat each time point as an independent observation (no temporal correlations).
- The goal is to infer the source signals, $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{\theta})$. $\boldsymbol{W}$ is called the **mixing matrix**. If $L = D$ (number of sources = number of sensors), it will be a square matrix.
- The model up to now is identical to factor analysis (except we don't in general require orthogonality of $\boldsymbol{W}$).
- However, we will use a different prior for $p(\boldsymbol{z}_t)$. In PCA, we assume each source is independent, and has a Gaussian distribution

$$p(\boldsymbol{z_t}) = \prod_{j=1}^{L} \mathcal{N}(z_{tj} | 0,1)$$

# Independent Component Analysis

- In ICA, we let the sources be non-Gaussian distributions.

$$p(\boldsymbol{z_t}) = \prod_{j=1}^{L} p_j(z_{tj})$$

- We constrain the variance of the source distributions to be 1 (any other variance can be modelled by scaling the rows of $\boldsymbol{W}$).
- The resulting model is known as **independent component analysis** or **ICA**.
- Using Gaussian distribution as a source prior in ICA does not permit unique recovery of the sources (shown in the next slide).
- This is because the PCA likelihood is invariant to any orthogonal transformation of the sources $\boldsymbol{z}_t$ and mixing matrix $\boldsymbol{W}$.
- PCA can recover the best linear subspace in which the signals lie, but cannot uniquely recover the signals themselves.

# Difficulties with the ICA

- Let us return to the ICA problem with independent i.i.d. sources

$$\boldsymbol{x} = \boldsymbol{W}\boldsymbol{z}, \boldsymbol{z} = [z_1, \ldots, z_L] \; are \; jointly \; independent$$

- We can easily see that the sources can be recovered only up to permutation, sign and scale. Indeed note the following:

$$\boldsymbol{x} = (\boldsymbol{W}\boldsymbol{P}^{-1}\boldsymbol{\Lambda}^{-1})(\boldsymbol{\Lambda}\boldsymbol{P}\boldsymbol{z})$$

where $\boldsymbol{P}$ is an arbitrary permutation matrix and $\boldsymbol{\Lambda}$ an arbitrary scaling matrix.

- In what follows we assume the following:

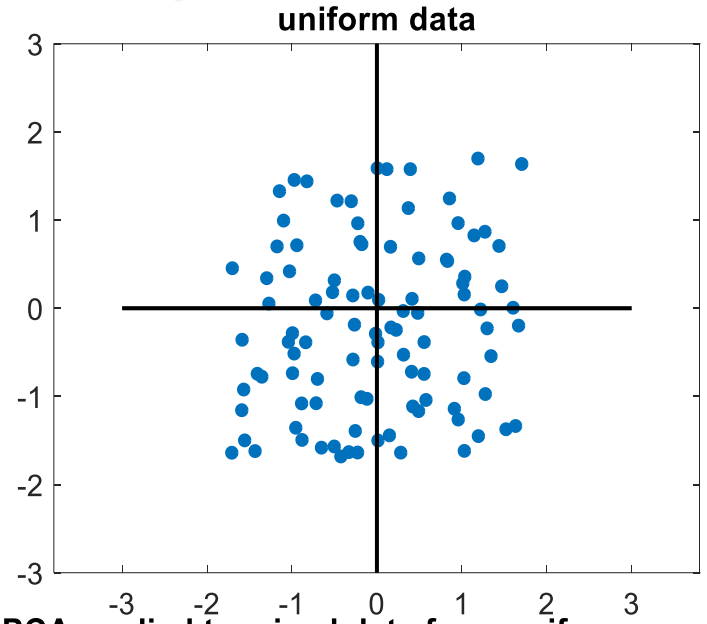
$$\mathbb{E}[\boldsymbol{z}] = 0, and \; \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^T] = \mathbf{I}$$

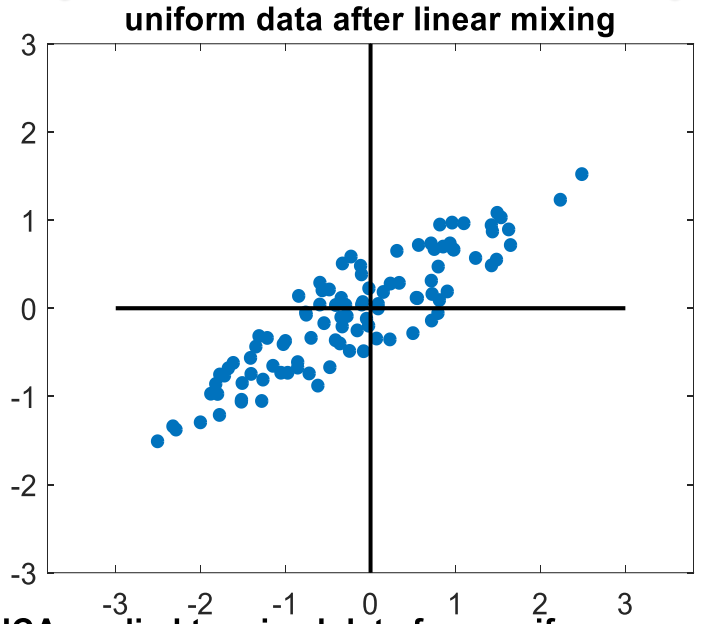
- The mixing matrix is not affected: $\mathbf{x} - \mathbb{E}[\boldsymbol{x}] = \boldsymbol{W}(\boldsymbol{z} - \mathbb{E}[\boldsymbol{z}])$

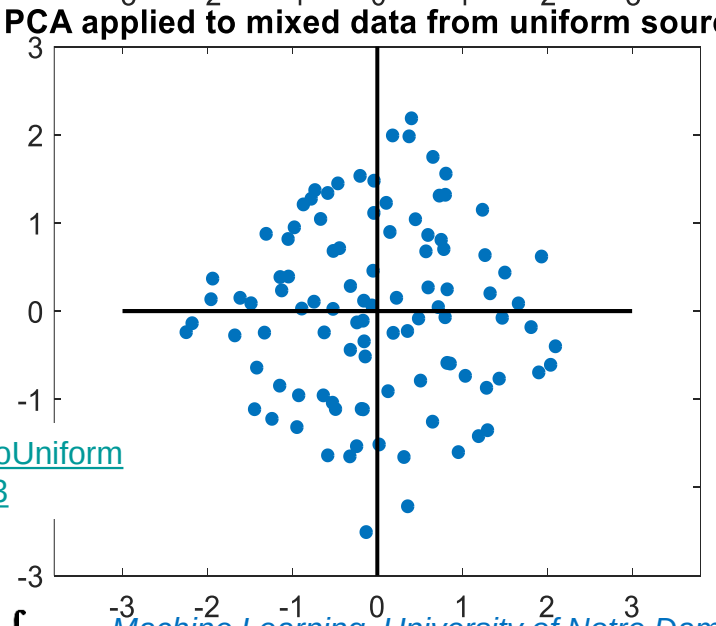
# Independent Component Analysis

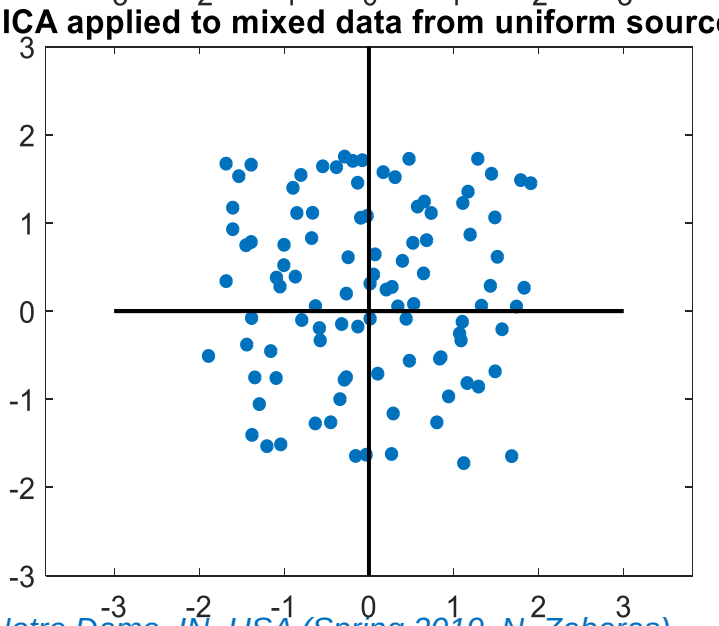
- Consider 2 independent sources with uniform PDFs with $\boldsymbol{W} = \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$.
- Then we observe the data shown after this linear mixing.
- If we apply PCA followed by scaling, we get results that correspond to a whitening of the data.
- To uniquely recover the sources, we need to perform an additional rotation.
- However there is no information in the symmetric Gaussian posterior to tell us which angle to rotate by.
- PCA solves "half" of the problem, since it identifies the linear subspace; all that ICA has to do is then to identify the appropriate rotation.
- ICA is similar to varimax which seek good rotations of the latent factors to enhance interpretability.

# Independent Component Analysis




Run icaDemoUniform
From PMTK3

# Independent Component Analysis

- ICA can recover the source, up to a permutation of the indices and possible sign change.
- ICA requires that $\boldsymbol{W}$ is square and hence invertible.
- In the non-square case (e.g., where we have more sources than sensors), we cannot uniquely recover the true signal, but we can compute the posterior $p(\boldsymbol{z_t}|\boldsymbol{x_t}, \widehat{\boldsymbol{W}})$.
- In both cases, we need to estimate $\boldsymbol{W}$ as well as the source distributions $p_j$ .

# Review of Fundamental Concepts

$$Y_1, Y_2 \text{ are independent } \leftrightarrow p(Y_1, Y_2) = p(Y_1)p(Y_2)$$

- Let $h_1, h_2$ be arbitrary functions. The following holds:

$$Y_1, Y_2 \text{ are independent } \rightarrow \mathbb{E}[h_1(Y_1)h_2(Y_2)] = \mathbb{E}[h_1(Y_1)]\mathbb{E}[h_2(Y_2)]$$

- With the following definition of correlation

$$corr(Y_1, Y_2) = \frac{\mathbb{E}[(Y_1 - \mathbb{E}[Y_1])(Y_2 - \mathbb{E}[Y_2])]}{\sqrt{var[Y_1]}\sqrt{var[Y_2]}}$$

the following holds:

$$corr(Y_1, Y_2) = 0 \leftrightarrow \mathbb{E}[Y_1 Y_2] = \mathbb{E}[Y_1]\mathbb{E}[Y_2]$$

- Let $Y_1, Y_2$ be jointly Gaussian. The following holds:

$$Y_1, Y_2 \text{ are independent } \leftrightarrow Y_1, Y_2 \text{ are uncorrelated}$$

# Review of Fundamental Concepts

- Recall the definition of the Shanon entropy:

$$H(Y_1, Y_2, \dots, Y_L) = -\int p(y_1, y_2, \dots, y_L) log(y_1, y_2, \dots, y_L) d\boldsymbol{y}$$

- The mutual information of a set of random variables is:

$$0 \leq I(Y_1, Y_2, \dots, Y_L) = \int p(y_1, y_2, \dots, y_L) log \frac{p(y_1, y_2, \dots, y_L)}{p(y_1) \dots p(y_L)} d\boldsymbol{y}$$

$$= KL\big(p(y_1, y_2, \dots, y_L) || p(y_1) \dots p(y_L)\big)$$

$$= \sum_{i=1}^{L} H(Y_i) - H(Y_1, Y_2, \dots, Y_L)$$

# Independent Component Analysis

- A possible approach is given in Mackay'03 (chapter 34) that does not consider the temporal aspect of the problem and treats the successive samples as i.i.d.

- Consider generative model with

  - the two latent variables: unobserved speech signal amplitudes
  - the two observed signal values $\boldsymbol{x} = [x_1\ x_2]^T$ at the microphones

- Distribution of latent variables factorizes as $p(\boldsymbol{z}) = p(z_1)p(z_2)$

- No need to include noise: observed variables = deterministic linear combinations of latent variables as

$$x = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

# Blind Source Separation: ICA

- Given a set of observations
  - the likelihood function is a function of the coefficients $W_{ij}$
  - log likelihood maximized using gradient-based optimization

  This is particular case of independent component analysis
- This requires that the latent variables have non Gaussian distributions
  - Probabilistic PCA: latent-space distribution = zero-mean isotropic Gaussian
  - No way to distinguish between two choices for the latent variables that differ by a rotation in the latent space
- PCA represents a rotation of the coordinate system in data space to diagonalize the covariance matrix. Zero correlation however does not imply independence!

# Maximum Likelihood Estimation

- Consider estimating square $\boldsymbol{W}$ for the noise-free ICA. We assume that the observations have been centered; hence we also assume $\boldsymbol{z}$ is zero-mean. In addition, we assume the observations have been whitened (done with PCA).

$$\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T] = \boldsymbol{I}$$

- For the noise free case we have:

$$cov(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T] = \boldsymbol{W}\mathbb{E}[\boldsymbol{z}z^T]\boldsymbol{W}^T = \boldsymbol{W}\boldsymbol{W}^T$$

- Hence we see that $\boldsymbol{W}$ must be orthogonal. This reduces the number of parameters we have to estimate from $D^2$ to $D(D-1)/2$.

- Let $\boldsymbol{V} = \boldsymbol{W}^{-1}$ (**recognition weights**), as opposed to $\boldsymbol{W}$ (**generative weights**)

- Since $\boldsymbol{x} = \boldsymbol{W}\boldsymbol{z}$, $p_x(\boldsymbol{W}\boldsymbol{z_t}) = p_z(\boldsymbol{z}_t)|\det(\boldsymbol{W}^{-1})| = p_z(\boldsymbol{z}_t)|\det(\boldsymbol{V})|$. Thus for $T$ iid samples ($\boldsymbol{v}_j$ being the $j$ –th row of $\boldsymbol{V}$)

$$\frac{1}{T}logp(\mathcal{D}|\boldsymbol{V}) = log|\det(\boldsymbol{V})| + \frac{1}{T}\sum_{j=1}^{L}\sum_{t=1}^{T} logp_j\left(\boldsymbol{v}_j^T\boldsymbol{x}_t\right)$$

# Maximum Likelihood Estimation

$$\frac{1}{T} logp(\mathcal{D}|\boldsymbol{V}) = log|\det(\boldsymbol{V})| + \frac{1}{T}\sum_{j=1}^{L}\sum_{t=1}^{T} logp_j\left(\boldsymbol{v}_j^T \boldsymbol{x}_t\right)$$

- Since we are constraining $\boldsymbol{V}$ to be orthogonal, the 1st term is a constant. We can also replace the average over the data with an expectation.

$$NLL(V) = \sum_{j=1}^{L} \mathbb{E}\left[G_j(\boldsymbol{z}_j)\right],\ \boldsymbol{z}_j = \boldsymbol{v}_j^T \boldsymbol{x}, G_j(z) = -logp_z(z)$$

- We want to minimize this subject to the constraint that the rows of $\boldsymbol{V}$ are orthogonal. We also want them to be unit norm, since this ensures that the variance of the factors is unity (with whitened data, $\mathbb{E}\left[(\boldsymbol{v}_j^T \boldsymbol{x})(\boldsymbol{x}^T \boldsymbol{v}_j)\right] = \left\|\boldsymbol{v}_j\right\|^2$ which is necessary to fix the scale of the weights. In otherwords, $\boldsymbol{V}$ should be an orthonormal matrix.
- Possible approaches include:
  - ✓ gradient descent algorithm to fit this model (rather slow);
  - ✓ natural gradient (faster) (MacKay 2003, ch 34);
  - ✓ Approximate Newton method; EM.

# Fast ICA Algorithm

- Define the following:

$$f(\boldsymbol{v}) = \mathbb{E}[G(\boldsymbol{v}^T\boldsymbol{x})] + \lambda(1 - \boldsymbol{v}^T\boldsymbol{v})$$
$$\nabla f(\boldsymbol{v}) = \mathbb{E}[\boldsymbol{x}g(\boldsymbol{v}^T\boldsymbol{x})] - \beta\boldsymbol{v}, \beta = 2\lambda$$
$$\boldsymbol{H}(\boldsymbol{v}) = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T g'(\boldsymbol{v}^T\boldsymbol{x})] - \beta\boldsymbol{I}$$

- Consider the following approximation:

$$\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T g'(\boldsymbol{v}^T\boldsymbol{x})] \approx \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T]\mathbb{E}[g'(\boldsymbol{v}^T\boldsymbol{x})] = \mathbb{E}[g'(\boldsymbol{v}^T\boldsymbol{x})]$$

- The following Newton update can now be introduced:

$$\boldsymbol{v}^* = \boldsymbol{v} - \frac{\mathbb{E}[\boldsymbol{x}g(\boldsymbol{v}^T\boldsymbol{x})] - \beta\boldsymbol{v}}{\mathbb{E}[g'(\boldsymbol{v}^T\boldsymbol{x})] - \beta} \text{ or } \boldsymbol{v}^* \triangleq \mathbb{E}[\boldsymbol{x}g(\boldsymbol{v}^T\boldsymbol{x})] - \mathbb{E}[g'(\boldsymbol{v}^T\boldsymbol{x})]\boldsymbol{v}$$

- The expectations can be replaced by Monte Carlo estimates from the training set (efficient online learning algorithm). After performing this update, one should project back onto the constraint surface using

$$\boldsymbol{v}^{new} = \frac{\boldsymbol{v}^*}{\|\boldsymbol{v}^*\|}$$

# Fast ICA Algorithm

- Define the following:

$$\boldsymbol{v}^{new} = \frac{\boldsymbol{v}^*}{\|\boldsymbol{v}^*\|}$$

- One iterates this algorithm until convergence.
  - ✓ Due to the sign ambiguity of $\boldsymbol{v}$, the values of $\boldsymbol{v}$ may not converge, but the direction defined by this vector should converge, so
  - ✓ can assess convergence by monitoring $|\boldsymbol{v}^T \boldsymbol{v}^{new}|$, which should approach $1$
- Since the objective is not convex, there are multiple local optima.

# Fast ICA Algorithm

- We can learn multiple different weight vectors or **features**.
  - We can either learn the features sequentially and then project out the part of $\boldsymbol{v}_j$ that lies in the subspace defined by earlier features, or
  - We can learn them in parallel, and orthogonalize $\boldsymbol{V}$ in parallel.
  - This latter approach is usually preferred, since, unlike PCA, the features are not ordered in any way (e.g. the 1st feature is not more important than the 2nd and hence it is better to treat them symmetrically).

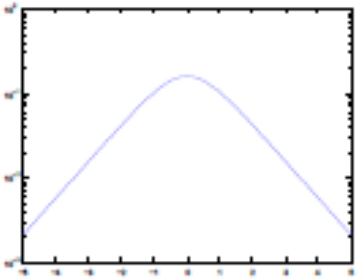
# Modeling the Source Densities

- So far, we have assumed that $G(z) = -\log p(z)$ is known.
- What kinds of models might be reasonable as signal priors?
- We know that using Gaussians (which correspond to quadratic functions $G$) won't work.
- So we want some kind of non-Gaussian distribution.
- In general, there are several kinds of non-Gaussian distributions, such as the following:
  - Super-Gaussian distributions: $\text{kurt}(z) > 0$,e.g., Laplace distribution.
  - Sub-Gaussian distribution: $\text{kurt}(z) < 0$. e.g., Uniform distribution.
  - Skewed distribution: Another way to "be non-Gaussian" is to be asymmetric (one possible measure of asymmetry is skewness). E.g., Gamma distribution (right skewed).

# Independent Component Analysis

- Common choice for latent-variable distribution:

$$p(z_j) = \frac{1}{\pi \cosh(z_j)} = \frac{2}{\pi\left(e^{z_j} + e^{-z_j}\right)}$$ (heavy tails)



- Original ICA model (Bell and Sejnowski, 1995) based on optimization of an objective function defined by information maximization.


- Bell, A. J. and T. J. Sejnowski (1995). An information maximization approach to blind separation and blind deconvolution. *Neural Computation* **7**(6), 1129–1159.

# Source Densities

- SuperGaussian (big spike at the mean, heavy tales, thin kurtosis): The Laplace distribution is an example. With $\mu_k$ denoting the $k^{\text{th}}$ moment of the distribution, we define:

$$kurt(z) = \frac{\mu_4}{\sigma^2} - 3 > 0$$

We subtract 3 from the standard definition to make the kurtosis of a Gaussian distribution equal to zero.

- SubGaussian (broad kurtosis): $kurt(z) < 0$

- Skewed distributions: The skewness is a measure of the degree of the asymmetry of the distribution:

$$\text{skew}(z) = \frac{\mu_3}{\sigma^3}$$


- Pham, D.-T. and P. Garrat (1997). Blind separation of mixture of independent sources through a quasimaximum likelihood approach. *IEEE Trans. on Signal Processing 45*(7), 1712–1725.


# Source Densities

- The empirical distribution of many natural signals (images, speech, when passed through certain linear filters) tend to be super-Gaussian.
- This holds for the kind of linear filters found in the brain (e.g. visual cortex), and in signal processing (wavelet transforms).
- One obvious choice for modeling natural signals with ICA is therefore the Laplace distribution. For mean zero and variance $1$, this has a log pdf given by

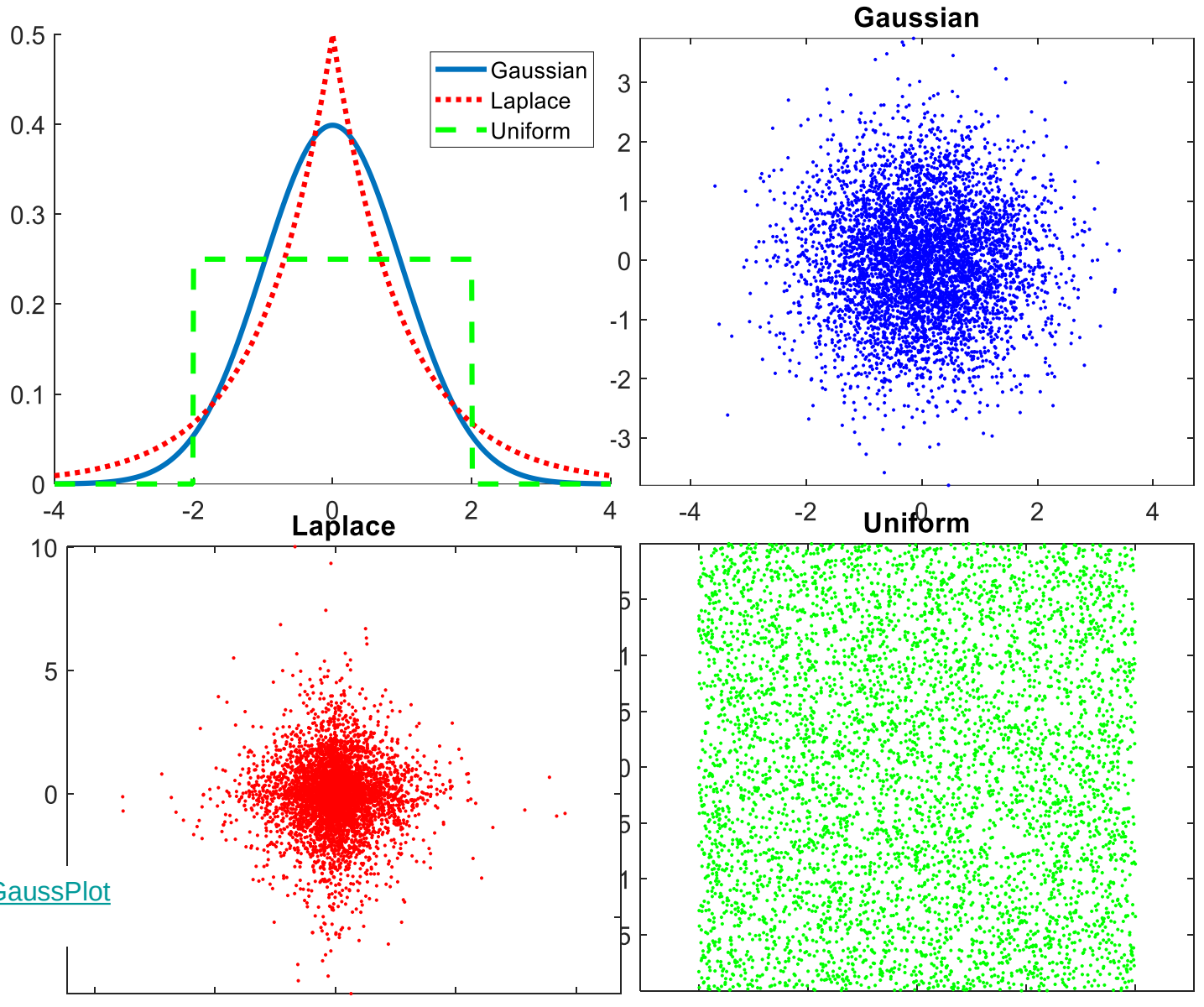
$$logp(z) = -\sqrt{2}|z| - \log(\sqrt{2})$$

- The logistic distribution is a smoother super-Gaussian distribution. For mean $0$ and variance $1$ (so $\mu = 0, s = \frac{\sqrt{3}}{\pi}$):

$$logp(z) = -2logcosh\left(\frac{\pi}{2\sqrt{3}}z\right) - log\left(\frac{4\sqrt{3}}{\pi}\right)$$

- As the exact type of source distribution is not that important, we often use: $\mathrm{G(z)} = -logp(z) = |z|$ $or$ $\mathrm{G(z)} = \mathrm{logcosh(z)}$.

# Source Densities




Run subSuperGaussPlot
From PMTK3

# Using EM

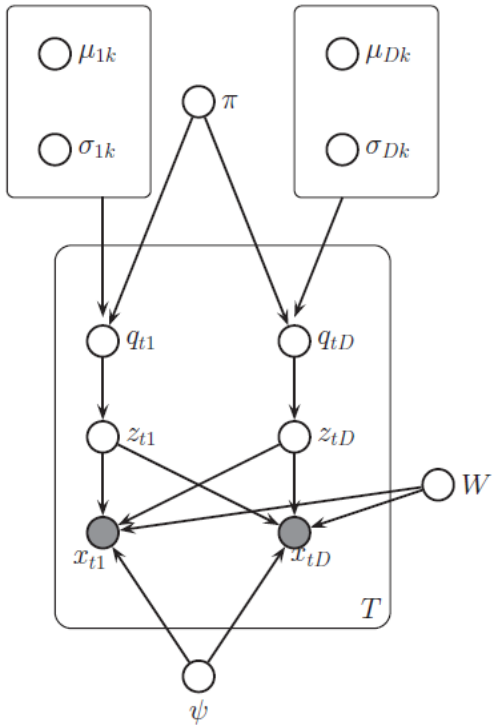
- An alternative to assuming a form for $p(\boldsymbol{z})$, is to use a mixture of (uni-variate) Gaussians:

$$p(q_j = k) = \pi_k$$
$$p(z_j | q_j = k) = \mathcal{N}(\mu_j, \sigma_{jk}^2)$$
$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{W}\boldsymbol{z}, \boldsymbol{\Psi})$$

- We can compute $\mathbb{E}[\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{\theta}]$ by summing over all $K^L$ combinations of the $\boldsymbol{q}_t$ variables, where $K$ is the number of mixture components per source (one can also use a mean field approximation)

- We can then estimate all the source distributions in parallel by fitting a standard GMM to $\mathbb{E}[\boldsymbol{z}_t]$. When the source GMMs are known, we can compute the marginals $p_j(z_j)$ very easily, using $p_j(z_j) = \sum_{k=1}^{K} \pi_{jk} \mathcal{N}(\mu_{jk}, \sigma_{jk}^2)$

- Given the $\pi_j$'s, we can then use an ICA algorithm to estimate $\boldsymbol{W}$.
- These steps should be interleaved.





- Attias, H. (1999a). Independent factor analysis. *Neural Computation* **11**(4), 803–851.
- Moulines, E., J.-F. Cardoso, and E. Gassiat (1997). Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'97)*, Munich Germany, pp. 3617–3620.

# Independent Component Analysis

- Independent factor analysis (*Attias*, 1999) considers a model in which the number of latent and observed variables can differ, the observed variables are noisy, and the individual latent variables have flexible distributions modeled by mixtures of Gaussians.
  - The log likelihood for this model is maximized using EM, and the reconstruction of the latent variables is approximated using a variational approach.
- References to other works are given next


- Attias, H. (1999a). Independent factor analysis. *Neural Computation* **11**(4), 803–851.


# Independent Component Analysis

- Some key works include:


- Hyvarinen, A. and E. Oja (1997). A fast fixed-point algorithm for independent component analysis. *Neural Computation* **9**(7), 1483–1492.
- Attias, H. (1999a). Independent factor analysis. *Neural Computation* **11**(4), 803–851.
- Jutten, C. and J. Herault (1991). Blind separation of sources, 1: An adaptive algorithm based on neuromimetic architecture. *Signal Processing* **24**(1), 1–10.
- Comon, P., C. Jutten, and J. Herault (1991). Blind source separation, 2: problems statement. *Signal Processing* **24**(1), 11–20.
- Amari, S., A. Cichocki, and H. H. Yang (1996). A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems*, Volume 8, pp. 757–763. MIT Press.
- Pearlmutter, B. A. and L. C. Parra (1997). Maximum likelihood source separation: a context-sensitive generalization of ICA. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems*, Volume 9, pp. 613–619. MIT Press.
- Hinton, G. E., M. Welling, Y. W. Teh, and S. Osindero (2001). A new view of ICA. In *Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation*, Volume 3.
- Miskin, J. W. and D. J. C. MacKay (2001). Ensemble learning for blind source separation. In S. J. Roberts and R. M. Everson (Eds.), *Independent Component Analysis: Principles and Practice*. Cambridge University Press.
- Hojen-Sorensen, P. A., O.Winther, and L. K. Hansen (2002). Mean field approaches to independent component analysis. *Neural Computation* **14**(4), 889–918.
- Choudrey, R. A. and S. J. Roberts (2003). Variational mixture of Bayesian independent component analyzers. *Neural Computation* **15**(1), 213–252.
- Chan, K., T. Lee, and T. J. Sejnowski (2003). Variational Bayesian learning of ICA with missing data. *Neural Computation* **15**(8), 1991–2011.
- Stone, J. V. (2004). *Independent Component Analysis: A Tutorial Introduction*. MIT Press.


# Other Estimation Principles

- It is quite common to estimate the parameters of ICA models using methods that seem different to maximum likelihood.
- We will review some of these methods below, because they give additional insight into ICA.
- However, we will also see that these methods in fact are equivalent to maximum likelihood after all.

# Maximizing non-Gaussianity

- Can pose the ICA problem as follows: Find $\boldsymbol{V}$ such that the distribution $\boldsymbol{z} = \boldsymbol{V}\boldsymbol{x}$ is as far from Gaussian as possible.

- One measure of non-Gaussianity is kurtosis, but this can be sensitive to outliers. Another measure is the **negentropy**, defined as

$$negentropy(z) = \mathbb{H}(\mathcal{N}(\mu, \sigma^2)) - \mathbb{H}(z), \mu = \mathbb{E}[z] \; and \; \sigma^2 = var[z].$$

- Since the Gaussian is the max entropy distribution, this measure is always non-negative and becomes large for distributions that are highly non-Gaussian. Using $\boldsymbol{z} = \boldsymbol{V}\boldsymbol{x}, \boldsymbol{V}$ orthogonal, whitten data (so $cov(\boldsymbol{z}) = \boldsymbol{I}$, independent of $\boldsymbol{V}$), we can define our objective as maximizing

$$J(\boldsymbol{V}) = \sum_j negentropy(z_j) = \sum_j \{\mathbb{H}(\mathcal{N}(\mu_j, \sigma_j^2)) - \mathbb{H}(z_j)\} = \sum_j -\mathbb{H}(z_j) + const$$

- Considering $\mathbb{H}(z_j) = -\mathbb{E}[\log p(z_j)]$, this gives the same form as maximizing the log-likelihood.


- Hyvarinen, A. and E. Oja (2000). Independent component analysis: algorithms and applications. *Neural Networks 13*, 411–430.

# Minimizing Mutual Information

- Since we are trying to find independent components, we can minimize the following:

$$I(\boldsymbol{z}) = KL\left(p(\boldsymbol{z})||\prod_j p(z_j)\right) = \sum_j -\mathbb{H}(z_j) - \mathbb{H}(\boldsymbol{z}) = -\sum_j \mathbb{H}(z_j) - \mathbb{H}(\boldsymbol{V}\boldsymbol{x})$$

- For $\boldsymbol{V}$ orthogonal, we can drop the last term, since then $\mathbb{H}(\boldsymbol{V}\boldsymbol{x}) = \mathbb{H}(\boldsymbol{x})$ (multiplying by $\boldsymbol{V}$ does not change the shape of the distribution), and $H(\boldsymbol{x})$ is a constant which is solely determined by the empirical distribution.

- Hence, we have $I(\boldsymbol{z}) = -\sum_j \mathbb{H}(z_j) + const.$

- Minimizing this is equivalent to maximizing the negentropy, which is equivalent to maximum likelihood.

# Maximizing Mutual Information

- Imagine a neural network where $\boldsymbol{x}$ is the input and $y_j = \phi(\boldsymbol{v}_j^T \boldsymbol{x}) + \varepsilon$ is the noisy output, $\phi$ is some nonlinear scalar function, and $\varepsilon \sim \mathcal{N}(0, 1)$.
- We want to maximize the mutual information between $\boldsymbol{y}$ (the internal neural representation) and $\boldsymbol{x}$ (the observed input signal). We have $\mathbb{I}(\boldsymbol{x}; \boldsymbol{y}) = \mathbb{H}(\boldsymbol{y}) - \mathbb{H}(\boldsymbol{y}|\boldsymbol{x})$, where the latter term is constant if we assume the noise has constant variance. One can show:

$$\mathbb{H}(\boldsymbol{y}) = \sum_{j=1}^{L} \mathbb{E}\left[log\phi'(\boldsymbol{v}_j^T \boldsymbol{x})\right] + log|\det(\boldsymbol{V})|$$

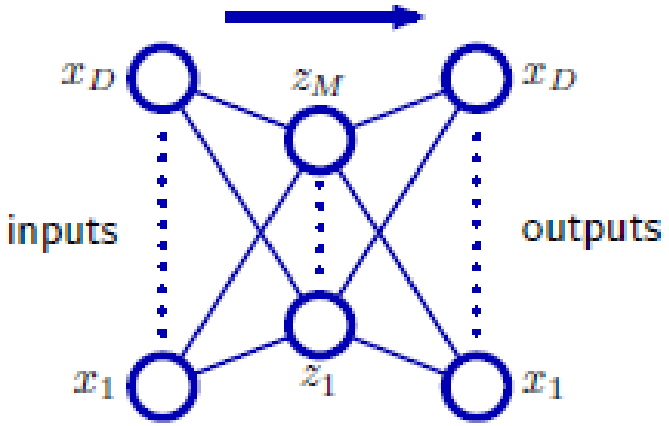
- We can drop the last term if $\boldsymbol{V}$ is orthogonal. If we define $\phi(\boldsymbol{z})$ to be a cdf, then $\phi'(\boldsymbol{z})$ is its pdf, and the above expression is equivalent to the log likelihood.
- If we use a logistic nonlinearity, $\phi(\boldsymbol{z}) = sigm(\boldsymbol{z})$, the corresponding pdf is the logistic distribution, and $\log \phi'(\boldsymbol{z}) = \log \cosh(\boldsymbol{z})$. Thus we see that infomax is equivalent to maximum likelihood.


- Bell, A. J. and T. J. Sejnowski (1995). An information maximisation approach to blind separation and blind deconvolution. *Neural Computation 7* (6), 1129–1159.

# Neural Networks for Dimensionality Reduction

# Autoassociative Neural Networks

- Neural networks for predicting outputs given inputs, can also be used for dimensionality reduction.

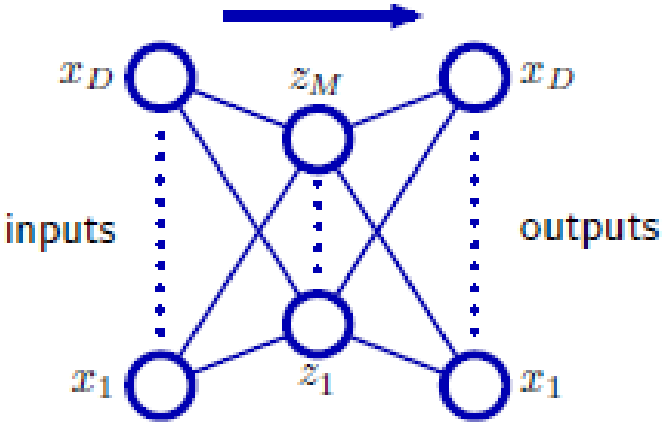



- We need a network having the same number of outputs as inputs, and optimize the weights to minimize the reconstruction error between inputs and outputs with respect to the training data set:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\| y(x_n, \boldsymbol{w}) - x_n \right\|^2$$

# Autoassociative Neural Networks

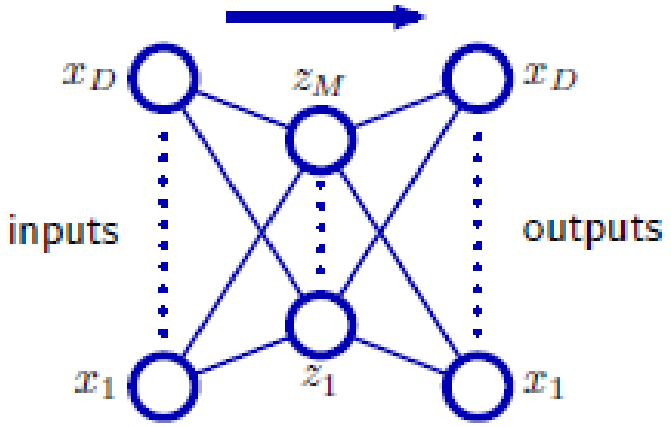
- Consider a multilayer perceptron: $D$ inputs, $D$ outputs and $M$ hidden units, with $M < D$.




- The targets used to train the network are the input vectors themselves i.e. the network is attempting to map each input vector onto itself (autoassociative mapping).
- Since $M < D$, a perfect reconstruction of all input vectors is not in general possible.
- Determine $\boldsymbol{w}$ by minimizing the mismatch between the input vectors and their reconstructions.

# Autoassociative Neural Networks

- Use the sum-of-square errors




$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\| y(x_n, \boldsymbol{w}) - x_n \right\|^2$$

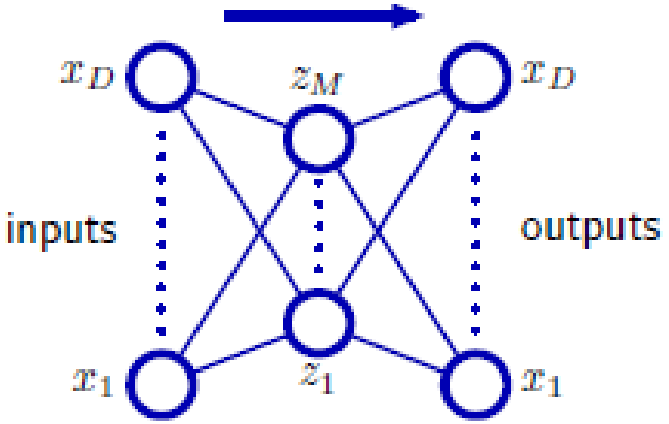
- For hidden units with linear activations functions,
  - the error function has a unique global minimum, and
  - at this minimum the network performs a projection onto the M-dimensional subspace which is spanned by the first $M$ principal components of the data.


- Bourlard, H. and Y. Kamp (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics* **59**, 291– 294.
- Baldi, P. and K. Hornik (1989). Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks* **2**(1), 53–58.

# Autoassociative Neural Networks

- Thus, the vectors of weights which lead into the hidden units form a basis set which spans the principal subspace. These vectors need not be orthogonal or normalized.

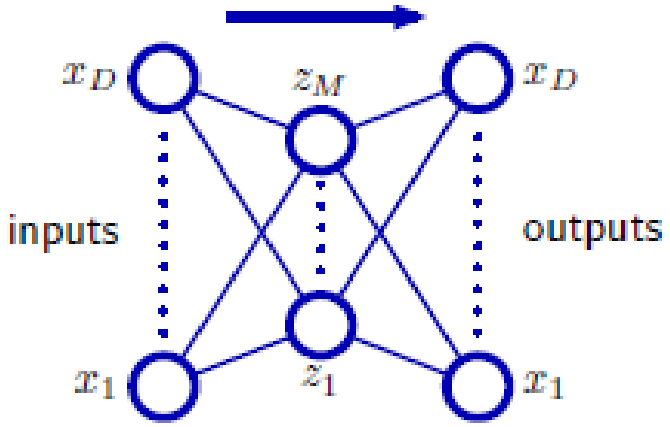
- Both PCA and the neural network use linear dimensionality reduction and are minimizing the same sum-of-squares error function → they both perform projection onto the $M$ –dimensional subspace spanned by the first $M$ –principal components of the data:




$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\|y(x_n, \boldsymbol{w}) - x_n\right\|^2$$

# Autoassociative Neural Networks

- Surprisingly, even with nonlinear hidden units in the two layers network below, the minimum error solution is again given by the projection onto the principal component subspace (Bourlard and Kamp, 1988).

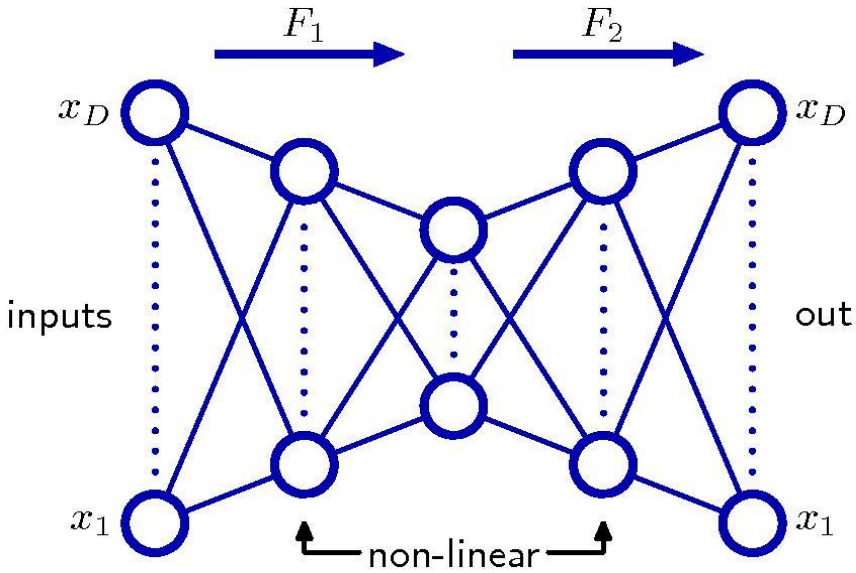



- There is therefore no advantage in using two layer neural networks to perform dimensionality reduction.
  - Standard PCA gives the correct solution in finite time, and generates ordered eigenvalues with orthonormal eigenvectors.


- Bourlard, H. and Y. Kamp (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics* **59**, 291– 294.

# Autoassociative Neural Networks

- Use more hidden layers (4 here) with some non-linear activation functions.

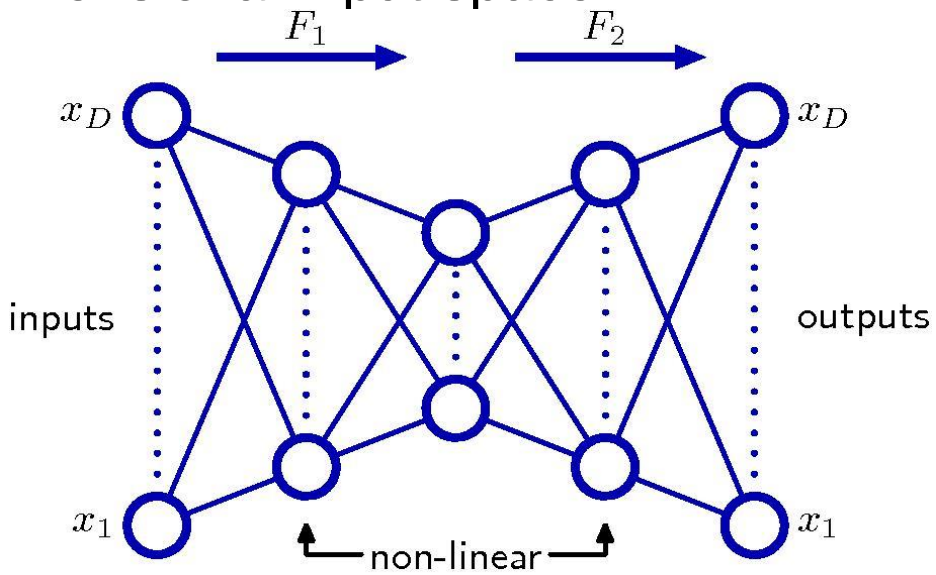



$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\|y(x_n, \boldsymbol{w}) - x_n\right\|^2$$

- The output units are linear, and the $M$ units in the 2nd hidden layer can also be linear. However, the 1st and 3nd hidden layers have sigmoidal nonlinear activation functions.
- Training involves minimization of the error function with risk of suboptimality.

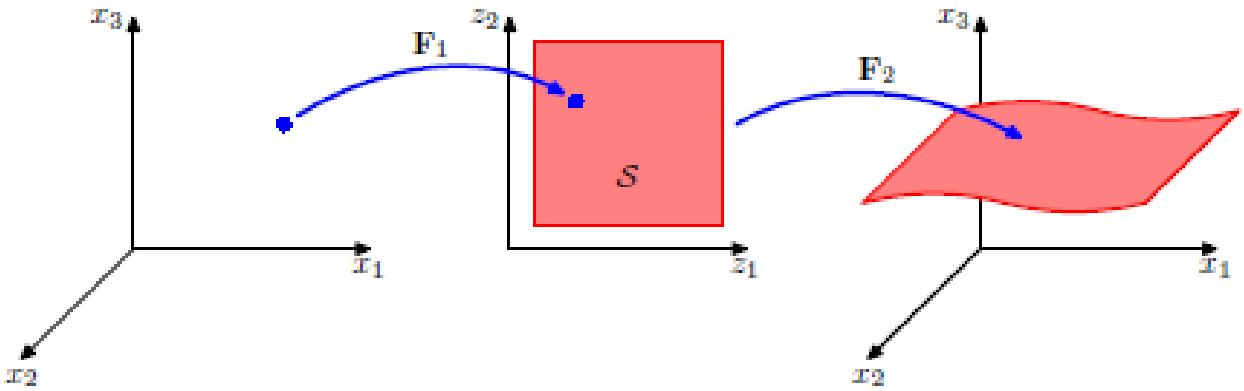
# Autoassociative Neural Networks

- The map $F_1$ projects the original $D$ dimensional data onto an $M$ −dimensional subspace $S$ defined by the activations of the units in the 2nd hidden layer.
- Because of the presence of the first hidden layer of nonlinear units, this mapping is not linear.
- Similarly, the 2nd half of the network defines an arbitrary functional mapping from the $M$ −dimensional space back into the original $D$ −dimensional input space.



# Nonlinear PCA with Autoassociative Neural Networks

- This has a simple geometrical interpretation. Consider $D = 3$ and $M = 2$.




- $F_2$ maps the $M$ −dimensional $S$ into a $D$ −dimensional space and therefore defines the way in which the space $S$ is embedded within the original $x$ −space.
- Since $F_2$ can be nonlinear, the embedding of $S$ can be nonplanar.
- $F_1$ defines a projection of points from the original $D$ −dimensional space into the $M$ −dimensional subspace $S$.

# Modeling Nonlinear Manifolds

# Modeling NonLinear Manifolds

- Data often lie in a manifold of lower-dimensionality than the observed data space
- Capturing this may improve the density modeling
- Possible approach: non-linear manifold modeled by piece-wise linear approximation, e.g.,
  - k-means + PCA for each cluster
  - better: use reconstruction error for cluster assignment (Kambhatla and Leen, 1997; Hinton *et al., 1997*)
- These are limited by not having an overall density model
- Tipping and Bishop: full probabilistic model using a mixture distribution in which components are probabilistic PCA
  - both discrete latent variables and continuous ones


- Kambhatla, N. and T. K. Leen (1997). Dimension reduction by local principal component analysis. *Neural Computation* **9**(7), 1493–1516.
- Hinton, G. E., P. Dayan, and M. Revow (1997). Modelling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks* **8**(1), 65–74.
- Tipping, M. E. and C. M. Bishop (1999a). Mixtures of probabilistic principal component analyzers. *Neural Computation* **11**(2), 443–482.

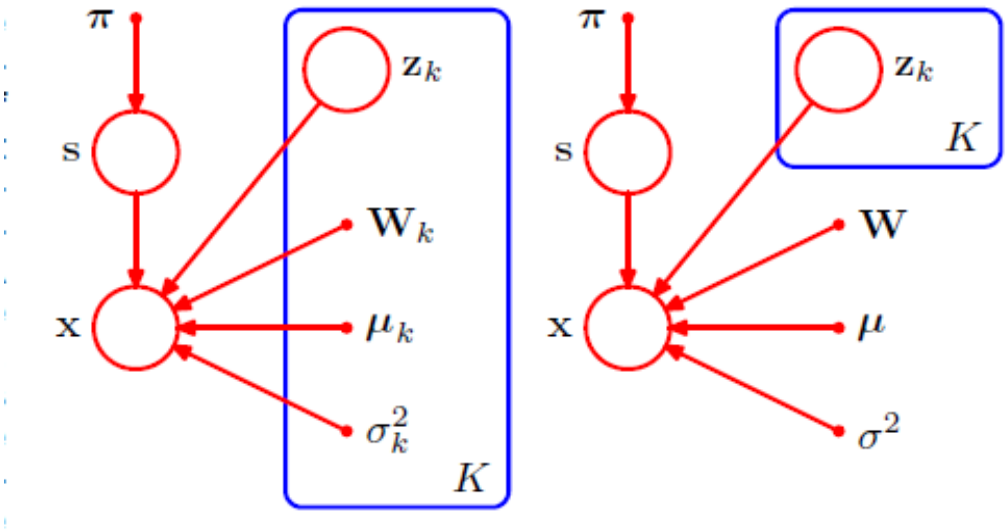

# Mixture of PCA Models

- A fully Bayesian treatment, based on variational inference (Bishop and Winn, 2000): both the number of components in the mixture and the effective dimensionalities of the individual models are inferred from the data.
- There are many variants of this model in which parameters such as the $\boldsymbol{W}$ matrix or the noise variances are tied across components in the mixture, or in which the isotropic noise distributions are replaced by diagonal ones, giving rise to a mixture of factor analyzers.
- Mixture of probabilistic PCA was extended hierarchically to produce an interactive data visualization algorithm.


- Bishop, C. M. and J. Winn (2000). Non-linear Bayesian image modelling. In *Proceedings Sixth European Conference on Computer Vision, Dublin*, Volume 1, pp. 3–17. Springer.
- Ghahramani, Z. and G. E. Hinton (1996a). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto.
- Ghahramani, Z. and M. J. Beal (2000). Variational inference for Bayesian mixtures of factor analyzers. In S. A. Solla, T. K. Leen, and K. R. Muller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12, pp. 449–455. MIT Press.
- Bishop, C. M. and M. E. Tipping (1998). A hierarchical latent variable model for data visualization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(3), 281–293.


# Mixture of PCA Models

- Graphical models for the mixture of probabilistic PCA.
- On the right, all model components share all of the parameters (not very useful model)
- $s$ is here the $k$ –nomial latent variable that selects the mixture components – It has a prior governed by parameter $\pi$.



# Principal Curves

- Alternative approach to use a single nonlinear model
- Principal Curves
  - Extension of PCA (that finds a linear subspace)
  - A curve is described by a vector-valued function $f(\lambda)$
  - Natural parametrization: the arc length along the curve
  - Given a point $\hat{x}$, we can find the closest point $\lambda = g_f(\boldsymbol{x})$ on the curve in terms of the Euclidean distance
  - A principal curve is a curve for which every point on the curve is the mean of all points in data space to project to it, so that

$$\mathbb{E}[\boldsymbol{x}|g_f(\boldsymbol{x}) = \lambda] = f(\lambda)$$

    There may be many principal curves for a continuous distribution
  - Hastie et al: two-stage iterative procedure for finding principal curves


- Hastie, T. and W. Stuetzle (1989). Principal curves. *Journal of the American Statistical Association* **84** (106), 502–516.


# Principal Curves

- For a given continuous density, there can be many principal curves.
- In practice, we are interested in finite data sets, and we also wish to restrict attention to smooth curves. Hastie and Stuetzle (1989) propose a two-stage iterative procedure for finding such principal curves, like the EM algorithm for PCA.
- The curve is initialized using the first principal component, and then the algorithm alternates between a data projection step and curve re-estimation step.
- Projection step: each data point is assigned to a value of $\lambda$ corresponding to the closest point on the curve.
- Re-estimation step: each point on the curve is given by a weighted average of those points that project to nearby points on the curve, with points closest on the curve given the greatest weight.


- Hastie, T. and W. Stuetzle (1989). Principal curves. *Journal of the American Statistical Association* **84** (106), 502–516.


# Principal Curves and Principal Surfaces

- When the subspace is constrained to be linear, the procedure converges to the first principal component
  - It is equivalent to the power method for finding the largest eigenvector of the covariance matrix.
- Principal curves can be generalized to multidimensional manifolds called principal surfaces
  - These have found limited use due to the difficulty of data smoothing in higher dimensions even for two-dimensional manifolds.

# Multidimensional Scaling

- PCA is often used for the purpose of visualization
- Another technique with a similar aim: multidimensional scaling (MDS, Cox and Cox 2000)
  - preserve as closely as possible the pairwise distances between data points.
  - involves finding the eigenvectors of the distance matrix.
  - equivalent results to PCA when the distance is Euclidean.
  - can be extended to a wide variety of data types specified in terms of a similarity matrix.


- Cox, T. F. and M. A. A. Cox (2000). *Multidimensional Scaling* (Second ed.). Chapman and Hall.

# Multidimensional Scaling

- Compute the matrix of squared pairwise similarities $\boldsymbol{D}$, $D_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$.
- Compute $\boldsymbol{J} = \boldsymbol{I}_N - 1/N\mathbf{1}\mathbf{1}^T$ (where $\boldsymbol{I}_N$ is the $N \times N$ identity function and $N$ is the number of datapoints).
- Compute $\boldsymbol{B} = -\frac{1}{2}\boldsymbol{J}\boldsymbol{D}\boldsymbol{J}^{\boldsymbol{T}}$.
- Find the $L$ largest eigenvalues $\lambda_i$ of $\boldsymbol{B}$, together with the corresponding eigenvectors $\boldsymbol{e_i}$.
- Put the eigenvalues into a diagonal matrix $\boldsymbol{V}$ and set the eigenvectors to be columns of matrix $\boldsymbol{P}$.
- Compute the embedding as $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{V}^{\mathbf{1/2}}$.

# Locally Linear Embedding

- Locally linear embedding (LLE, Roweis and Saul 2000).
- Compute the set of coefficients that best reconstruct each data point from its neighbors.
- Coefficients arranged to be invariant to rotation, translations, scaling

  Characterize the local geometrical properties of the neighborhood.
- LLE maps the high-dimensional data to a lower dimensional subspace while preserving these coefficients.
- These weights are used to reconstruct the data points in low-dimensional space as in the high-dimensional space.
- Albeit non linear, LLE does not exhibit local minima.


- Roweis, S. and L. Saul (2000, December). Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**, 2323–2326.

# Locally Linear Embedding

- Decide on the neighbours of each point (e.g., $K$ nearest neighbours):
  - compute distances between every pair of points
  - find the *k* smallest distances
  - set $\boldsymbol{W}_{ij} = 0$ for other points
  - for each point $\boldsymbol{x}_i$:
    - create a list of its neighbours' locations $\mathbf{z}_i$
    - compute $\boldsymbol{z}_i = \boldsymbol{z}_i - \boldsymbol{x}_i$
- Compute the weights matrix $\boldsymbol{W}$ that minimizes $\varepsilon = \sum_{i=1}^{N}\left(\boldsymbol{x}_i - \sum_{j=1}^{N} \boldsymbol{W}_{ij}\boldsymbol{x}_j\right)^2$ according to the constraints:
  - compute local covariance $\mathbf{C} = \mathbf{Z}\mathbf{Z}^T$ , where $\boldsymbol{Z}$ is the matrix of $\boldsymbol{z}_i$'s
  - solve $\boldsymbol{C}\boldsymbol{W} = \boldsymbol{I}$ for $\boldsymbol{W}$, where $\boldsymbol{I}$ is the $N \times N$ identity matrix
  - set $\boldsymbol{W}_{ij} = 0$ for non-neighbours
  - set other elements to $\boldsymbol{W}/\sum \boldsymbol{W}$
- Compute the lower dimensional vectors $\boldsymbol{y}_i$ that minimise $\boldsymbol{y}_i = \sum_{i=1}^{N}\left(\boldsymbol{y}_i -$

# ISOMAP

- Isometric feature mapping (ISOMAP, Tenenbaum et al. 2000)
- Goal: data projected to a lower-dimensional space using MDS

  but dissimilarities defined in terms of the geodesic distances on the manifold
- Algorithm:
  - First defines the neighborhood using KNN
  - Construct a neighborhood graph with weights corresponding to the Euclidean distances
  - Geodesic distance approximated by the sum of Euclidean distances along the shortest path connecting two points
  - Apply MDS to the geodesic distances


- Tenenbaum, J. B., V. de Silva, and J. C. Langford (2000, December). A global framework for nonlinear dimensionality reduction. *Science* **290**, 2319–2323.

# Latent Trait Models

- We can also consider models having continuous latent variables together with discrete observed variables, giving rise to latent trait models (Bartholomew, 1987).

- In this case, the marginalization over the continuous latent variables, even for a linear relationship between latent and observed variables, cannot be performed analytically, and so more sophisticated techniques are required.

- Tipping (1999) uses variational inference in a model with a two-dimensional latent space, allowing a binary data set to be visualized analogously to the use of PCA to visualize continuous data.


- Bartholomew, D. J. (1987). *Latent Variable Models and Factor Analysis*. Charles Griffin.
- Tipping, M. E. (1999). Probabilistic visualisation of high-dimensional binary data. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems*, Volume 11, pp. 592–598. MIT Press.


# Latent Trait Models

- This model is the dual of the Bayesian logistic regression problem.

- In the case of logistic regression, we have $N$ observations of the feature vector $\boldsymbol{\varphi}_n$ which are parametrized by a single parameter vector $\boldsymbol{w}$, whereas in the latent space visualization model there is a single latent space variable $\boldsymbol{x}$ (analogous to $\boldsymbol{\varphi}$) and $N$ copies of the latent variable $\boldsymbol{w}_n$.

- A generalization of probabilistic latent variable models to general exponential family distributions is described in Collins et al. (2002).


  - Collins, M., S. Dasgupta, and R. E. Schapire (2002). A generalization of principal component analysis to the exponential family. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems*, Volume 14, pp. 617–624. MIT Press.

# Density Network

- An arbitrary distribution can be formed by taking a Gaussian random variable and performing a nonlinear transformation.
- This is exploited in a latent variable model called a density network (MacKay, 1995; MacKay and Gibbs, 1999). The nonlinear function is governed by a multilayered neural net.
- If the network has enough hidden units, it can approximate a given nonlinear function to any desired accuracy.
- The marginalization over the latent variables to obtain the likelihood function, is no longer analytically tractable.
- The likelihood is approximated using Monte Carlo drawing samples from the Gaussian prior. The marginalization over the latent variables becomes a sum with one term for each sample.
- The procedure is costly: A large number of sample points may be required.


- MacKay, D. J. C. (1995). Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research, A* **354**(1), 73–80.
- MacKay, D. J. C. and M. N. Gibbs (1999). Density networks. In J. W. Kay and D. M. Titterington (Eds.), *Statistics and Neural Networks: Advances at the Interface*, Chapter 5, pp. 129–145. Oxford University Press.


# Generative Topographic Mapping

# Generative Topographic Mapping

- If we consider more restricted forms for the nonlinear function, and make an appropriate choice of the latent variable distribution, then we can construct a latent variable model that is both nonlinear and efficient to train.

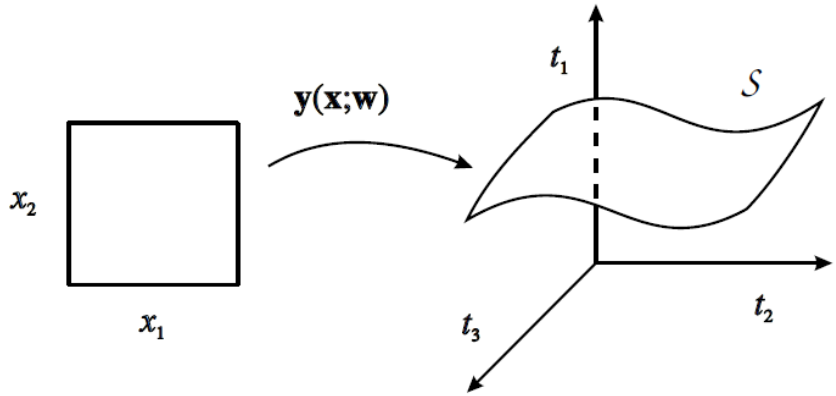
- The generative topographic mapping, or GTM (Bishop et al., 1996; Bishop et al., 1997; Bishop et al., 1998) uses a latent distribution that is defined by a finite regular grid of delta functions over the (typically two-dimensional) latent space.

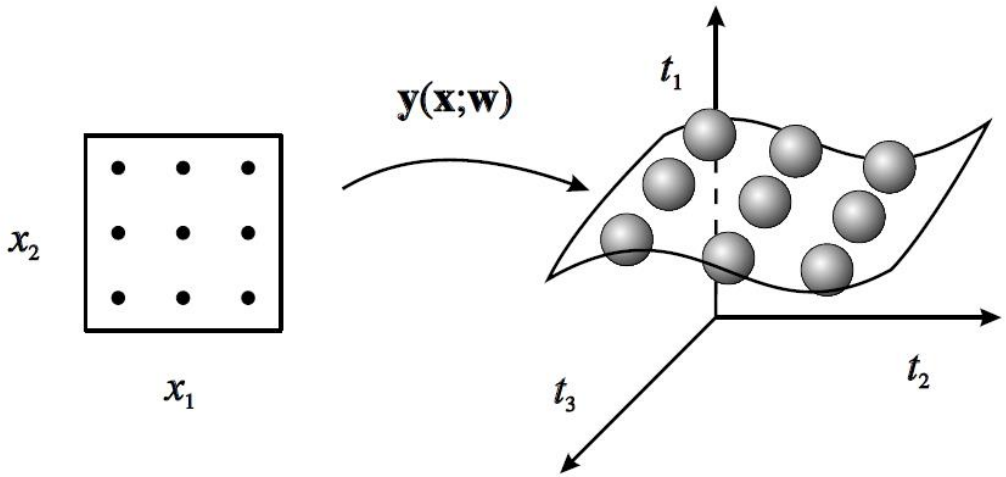
- Marginalization over the latent space then simply involves summing over the contributions from each of the grid locations.


- Bishop, C. M., M. Svens´en, and C. K. I. Williams (1997a). GTM: a principled alternative to the Self-Organizing Map. In M. C. Mozer, M. I. Jordan, and T. Petche (Eds.), *Advances in Neural Information Processing Systems*, Volume 9, pp. 354–360. MIT Press.
- Bishop, C. M., M. Svensen, and C. K. I. Williams (1998b). GTM: the Generative Topographic Mapping. *Neural Computation* **10**(1), 215–234.
- Bishop, C. M., M. Svensen, and C. K. I. Williams (1998a). Developments of the Generative Topographic Mapping. *Neurocomputing* **21**, 203–224.


# Generative Topographic Mapping

- The non-linear function $\boldsymbol{y}(\boldsymbol{x};\boldsymbol{W})$ defines a manifold $S$ embedded in data space given by the image of the latent-variable space under the mapping $\boldsymbol{x} \rightarrow \boldsymbol{y}$.



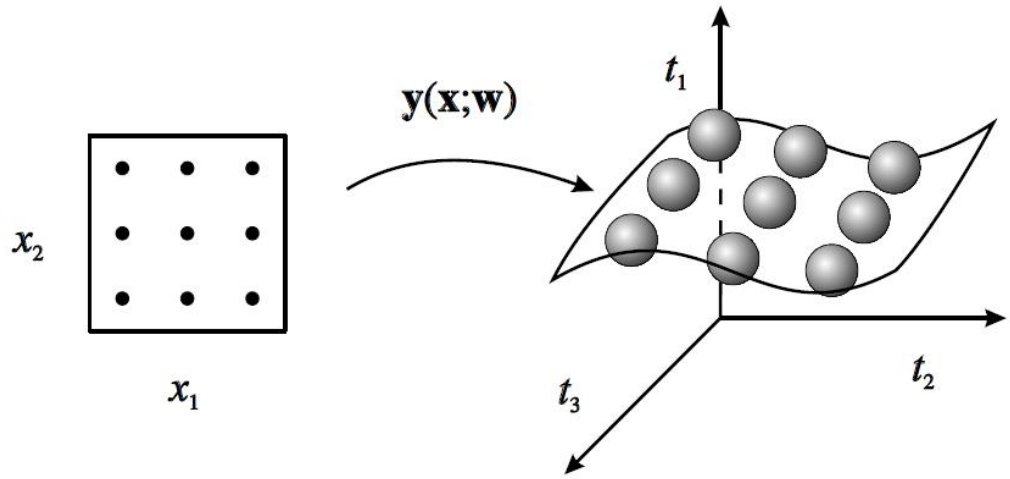

$$p(\boldsymbol{x}) = \frac{1}{K}\sum_{i=1}^{K}\delta(\boldsymbol{x}-\boldsymbol{x}_i)$$

$$p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{W},\beta) = \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left(-\frac{\beta}{2}\|\boldsymbol{y}($$

- We consider a prior distribution $p(\boldsymbol{x})$ consisting of a superposition of delta functions, located at the nodes of a regular grid in latent space. Each node $\boldsymbol{x}$ is mapped to a corresponding point $\boldsymbol{y}(\boldsymbol{x};\boldsymbol{W})$ in data space, and forms the center of a corresponding Gaussian distribution.

# Generative Topographic Mapping

- GTM results in a constrained Gaussian mixture model since the centers of the Gaussians cannot move independently but are constrained by $\boldsymbol{y}(\boldsymbol{x};\boldsymbol{W})$.




$$p(\boldsymbol{x}) = \frac{1}{K}\sum_{i=1}^{K}\delta(\boldsymbol{x}-\boldsymbol{x}_i)$$

$$p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{W},\beta) = \left(\frac{\beta}{2\pi}\right)^{D/2}\exp\left(-\frac{\beta}{2}\|\boldsymbol{y}(\boldsymbol{x};\boldsymbol{W})-\boldsymbol{t}\|^2\right)$$

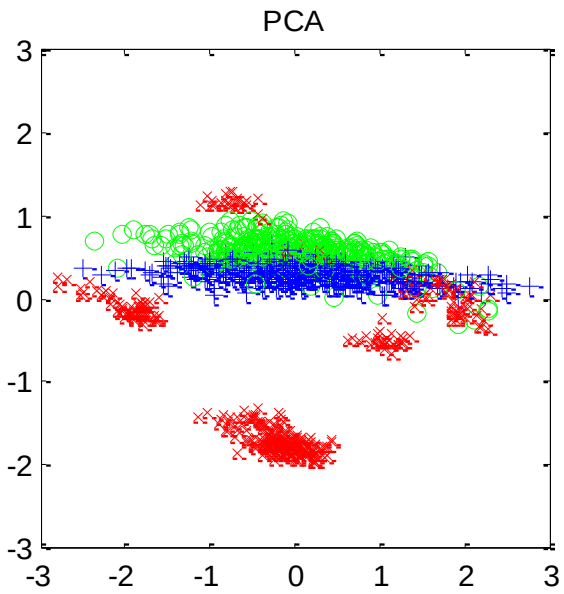
$$p(\boldsymbol{t}|\boldsymbol{W},\beta) = \int p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{W},\beta)p(\boldsymbol{x})d\boldsymbol{x} = \frac{1}{K}\sum_{i=1}^{K}p(\boldsymbol{t}_i|\boldsymbol{x}_i,\boldsymbol{W},\beta)$$

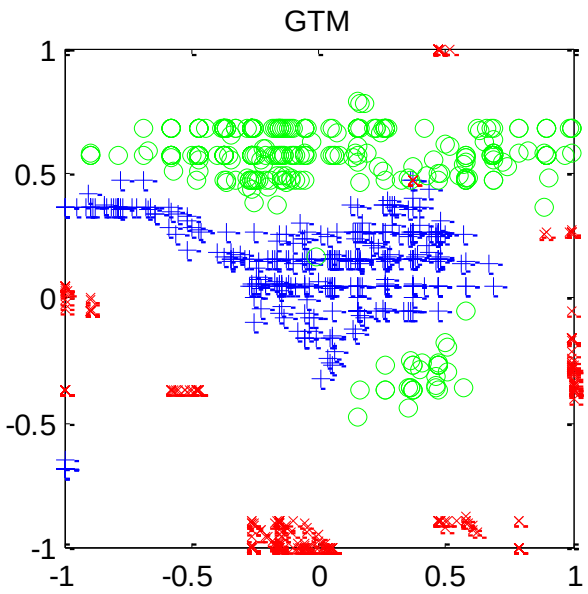
# Generative Topographic Mapping

- The nonlinear mapping is given by a linear regression model that allows for general nonlinearity while being a linear function of the adaptive parameters.
- The limitation of linear regression models arising from the curse of dimensionality does not arise in the context of the GTM since the manifold generally has 2 dimensions irrespective of the dimensionality of the data space.
- As a consequence of these two choices, the likelihood function can be expressed analytically in closed form and can be optimized efficiently using the EM algorithm.
- The resulting GTM model fits a 2D nonlinear manifold to the data set, and by evaluating the posterior distribution over latent space for the data points, they can be projected back to the latent space for visualization purposes.

# Generative Topographic Mapping

- The Figure shows a comparison of the oil data set visualized with linear PCA and with the nonlinear GTM.




Matlab Implementation

- Each data point is plotted at the mean of its posterior distribution in latent space. The nonlinearity of the GTM allows clear separation between the groups of data points.

# Self Organizing Maps (SOM)

- GTM can be seen as a probabilistic version of the self organizing map, which also represents a 2D nonlinear manifold as a regular array of discrete points.
- SOM is reminiscent of K-means in that data are assigned to nearby prototype vectors that are subsequently updated.
- Initially, the prototypes are distributed at random, and during the training process they 'self organize' so as to approximate a smooth manifold.
- Unlike K-means, the SOM is not optimizing any well-defined cost function (Erwin et al., 1992) making it difficult to set the parameters of the model and to assess convergence.
- There is no guarantee that the 'self-organization' will take place as this is dependent on the choice of appropriate parameter values for any particular data set.


- Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics* **43**, 59–69.
- Kohonen, T. (1995). *Self-Organizing Maps*. Springer.
- Erwin, E., K. Obermayer, and K. Schulten (1992). Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics* **67**, 47–55


# Self Organizing Maps

- A set of K reference vectors $\boldsymbol{z}_i$ is defined in the data space, in which each vector is associated with a node on a regular lattice in a 2D `feature map' (analogous to the latent space of GTM).

- The algorithm begins by initializing the reference vectors (random values, equal to a random subset of the data points, or by using PCA). Each cycle of the algorithm then proceeds as follows.

- For every $\boldsymbol{t}_n$ the corresponding `winning node' $j(n)$ is identified, corresponding to $\boldsymbol{z}_j$ having the smallest Euclidean distance $\|\boldsymbol{z}_j - \boldsymbol{t}_n\|^2$. The reference vectors are then updated by setting them equal to weighted averages of the data points given by ($h_{ij(n)}$ being a neighbourhood (Gaussian) function associated with node $i$)

$$\boldsymbol{z}_i = \frac{\sum_n h_{ij(n)} \boldsymbol{t}_n}{\sum_n h_{ij(n)}}$$

- The width of $h_{ij(n)}$ starts with a relatively large value and is gradually reduced after each iteration.

# Extensions of the GTM Model

- By contrast, GTM optimizes the log likelihood function, and the resulting model defines a probability density in data space.
- In fact, it corresponds to a constrained mixture of Gaussians in which the components share a common variance, and the means are constrained to lie on a smooth two-dimensional manifold.
- This probabilistic foundation also makes it very straightforward to define generalizations of GTM (Bishop *et al., 1998*) such as
  - a Bayesian treatment,
  - dealing with missing values,
  - a principled extension to discrete variables,
  - use of Gaussian processes to define the manifold, or
  - a hierarchical GTM model (Tino and Nabney, 2002).


- Tino, P. and I. T. Nabney (2002). Hierarchical GTM: constructing localized non-linear projection manifolds in a principled way. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(5), 639–656.
- Bishop, C. M., M. Svensen, and C. K. I. Williams (1998a). Developments of the Generative Topographic Mapping. *Neurocomputing* **21**, 203– 224.


# Extensions of the GTM Model

- Because the manifold in GTM is defined as a continuous surface, not just at the prototype vectors as in the SOM, it is possible to compute the magnification factors corresponding to the local expansions and compressions of the manifold needed to fit the data set (Bishop et al., 1997) as well as the directional curvatures of the manifold (Tino et al., 2001).

- These can be visualized along with the projected data and provide additional insight into the model.


- Tino, P. and I. T. Nabney (2002). Hierarchical GTM: constructing localized non-linear projection manifolds in a principled way. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(5), 639–656.
- Bishop, C. M., M. Svensen, and C. K. I. Williams (1998a). Developments of the Generative Topographic Mapping. *Neurocomputing* **21**, 203– 224.